MYD-SAM9G15/9G25 /9G35/9X25/9X35-V2 User Manual

Version V1.0

Version History

Version Number	Description	Time
V1.0	Initial Version	2013.11.4

Content

Content **1**

Chapter 1 Product Overview **1**

1.1 Product Description 1

1.2 Product Overview 1

1.3 Product Features 2

1.4 Products Packing 5

Chapter 2 Hardware Resource Introduction **6**

2.1 CPU module & Base Board Resources Overview 6

2.2 CPU Module Introduction 7

2.2.1 CPU Module Preview 7

2.2.2 CPU Performance 8

2.2.3 CPU Module Board Port definition 8

2.2.4 LED 10

2.2.5 Switch Setting 10

2.3 Base Board Introduction 11

2.3.1 Base Board Dimensions 11

2.3.2 Universal Serials 12

2.3.3 CAN BUS 13

2.3.4 JTAG Port 14

2.3.5 LCD Port 14

2.3.6 User Ports 15

2.3.7 Audio Module WM8731 15

2.3.8 USB Module 16

2.3.9 RS485 Bus 18

2.3.10 External UART 19

2.3.11 External RTC 20

2.3.12 ETHERNET 21

2.4 Jumper Settings 21
Chapter 3 MDK Example Code 23
3.1 Overview 23
3.2 Preparation 23
3.2.1 Configure And Compile 23
3.2.2 MDK Routine Debug 28
3.2.3 Super Terminal Configuration 30
3.2.4 Manual Download 34
3.2.5 Automatic Download 37
3.3 MDK Routine Introduction 38
3.3.1 Getting-started 38
3.3.2 Adc_adc10 39
3.3.3 Adc_touch-screen 40
3.3.4 CAN 41
3.3.5 DMA 43
3.3.6 LCD 44
3.3.7 Periph _protect 45
3.3.8 Pmc_clock_switching 47
3.3.9 PWM 49
3.3.10 Ssc_dma_audio 50
3.3.11 Twi_eeprom 52
3.3.12 Usart_serial 53
3.3.13 Emac0 54
3.3.14 Emac1 55
3.3.15 Hsmci_multimedia_card 56
3.3.16 Hsmci_sdcard 57
3.3.17 Smc_nandflash 58
3.3.18 Spi_serialflash 60
3.3.19 Usb_audio_looprec 61
3.3.20 Usb_cdc_serial 62

3.3.21 Usb_core 63
3.3.22 Usb_hid_key board 64
3.3.23 Usb_hid_mouse 65
3.3.24 Usb_hid_msd 66
3.3.25 Usb_hid_transfer 69
3.3.26 Usb_iad_cdc_cdc 71
3.3.27 Usb_iad_cdc_hid 72
3.3.28 Usb_iad_cdc_msd 74
3.3.29 Usb_massstorage 76
3.3.30 RS485 78
3.3.31 External GPIO 79
3.3.32 External UART 80
3.3.33 External RTC 80
Chapter 4 Linux System Guide 82
4.1 Outline 82
4.2 Software Resources 83
4.3 Start Linux System 84
4.3.1 Install Download Tool 84
4.3.2 Connect the Board to PC 84
4.3.3 Automatic Download 85
4.3.4 Manual Download 85
4.4 Linux Development Environment Structure 91
4.5 Installation And Compile 91
4.5.1 Create a Working Directory 91
4.5.2 Install Cross Compiler Tools 92
4.5.3 Install AT91Bootstrap Source and Compile 92
4.5.4 Install U-boot Source and Compile 92
4.5.5 Install and Compile Linux Kernel Source Code 93
4.6 Make Linux File System 94
4.6.1 Write Demo Program “helloworld” 94

4.6.2 Mount UBIFS File System 95
4.6.3 Modify UBIFS System Files 96
4.6.4 Regenerate UBIFS System File 96
4.7 Linux Application 99
4.7.1 Touch Screen Calibration 99
4.7.2 U-disk Use 100
4.7.3 SD Card Use 100
4.7.4 Play MP3 Music 101
4.7.5 Network Port Test 102
4.7.6 Telnet Test 103
4.7.7 Login Development Board By Telnet 104
4.7.8 Mount NFS File System 105
4.7.9 RTC Use 106
4.7.10 Serial Port 106
4.7.11 External GPIO 107
4.8 Linux Driver Development Example 108
4.8.1 Hardware Schematic 108
4.8.2 Driver Source Code 108
4.8.3 Compile Driver 112
4.8.4 Download Driver 113
4.9 Application Development Example 113
4.9.1 Source Code Compilation 113
4.9.2 Compile 115
4.9.3 Application Use 115
4.10 Qt Guide 115
4.10.1 Qt Cross Tool Chain 116
4.10.2 Establish Qt Development Environment 119
Chapter 5 Android System Guide 123
5.1 Overview 123
5.2 Software Resources 124

5.3 Build Android System 124
5.3.1 Install Download Tool 125
5.3.2 Connect the Board To SAM-BA 125
5.3.4 Manual Download 126
5.4 Compile Android System Files 135
5.4.1 Android System Principle 135
5.4.2 Compile System Files 136
5.5 Android System Application 137
5.5.1 USB Key Board Test 137
5.5.2 Browse Picture Test 137
5.5.3 Play Audio Test 139
5.5.4 Ethernet Test 140
Appendix 1 FAQ 144
Appendix 2 sales FAQ and technical support 148

Chapter 1 Product Overview

1.1 Product Description

MYIR has launched MYD-SAM9X5-V2 series boards which are based on Atmel AT91SAM9X5 series processor (AT91SAM9G15/25/35 and AT91SAM9X25/X35, are all based on the ARM926EJ-S kernel). MYD-SAM9X5-V2 series which run at up to 400 MHz have 256MB NandFlash, 4MB DataFlash, 128MB DDR2 SDRAM and support Linux 2.6.39 as well as Android 2.3.5 operating system. MYD-SAM9X5-V2 series also provide relevant sources and have rich peripheral Ports: High-speed USB2.0, Audio input, Audio output, LCD Port, CAN Port, 10/100Mbps Ethernet MAC, JTAG debug Port, Serial Port and Micro SD card Port.

1.2 Product Overview

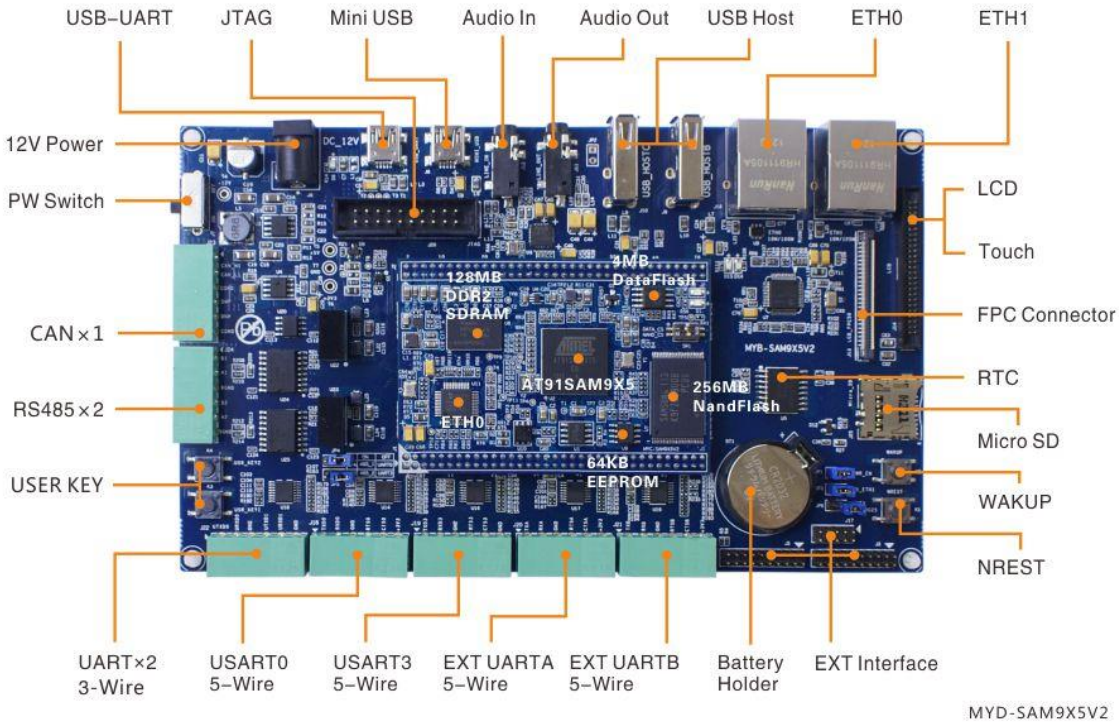


Figure 1-1

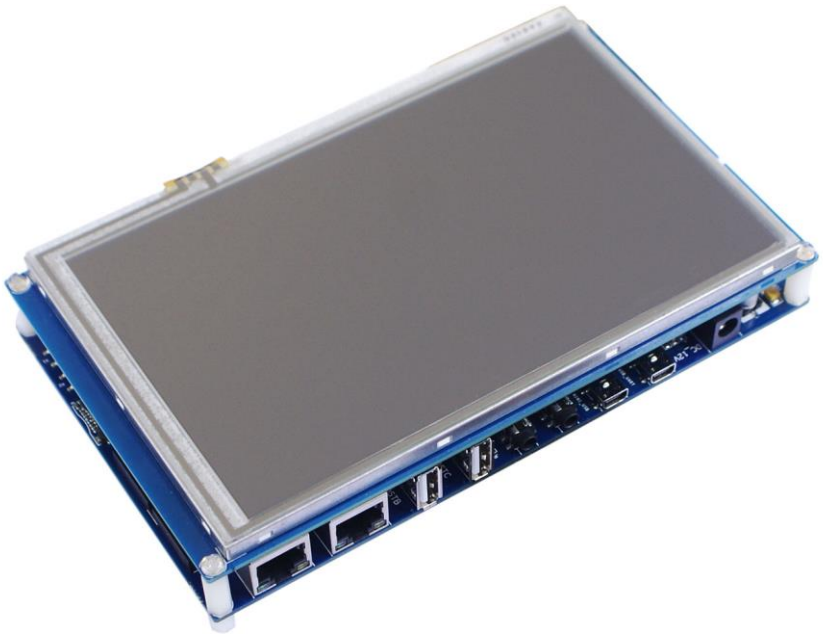


Figure 1-2

1.3 Product Features

The MYD-SAM9X5-V2 series are based on AT91SAM9X5 processor and inherit all its components and periphery Ports, its key features are as follows:

- Rich Peripherals

  Include Ethernet, USB2.0 Device, USART, SD Card and so on.

- High-performance Data Channel

  Atmel SAM9X5 series microprocessors which run at up to 400MHz have 12-layer bus matrix based high data bandwidth channel.

- Next-generation Memory

  LPDDR/DDR2 support ensures supply and cost efficiency. In addition, the MLC/SLC NAND Flash controller of MPU supports 24-bit Error Code Correction.

- Low Power and Low System Cost

  In backup mode, power consumption is only 300uW/MHz working at 400MHz and 8uA.3.3V IOs eliminates the need for external level shifter, while 0.8mm ball pitch packages reduce PCB design complexity and cost.

Basic features of MYD-SAM9X5-V2 series are listed as the following:

Electrical Parameters

- Operating Temperature:
  - Industrial Grade: -40 ℃ ~ +85 ℃
  - Commercial Grade: 0 ℃ ~ +70 ℃
- Electrical Specifications:
  - Base Board: 12V/1.25A
  - CPU Module: 3.3V/1A
- Mechanical Dimensions:
  - Base Board: 177mm x 106mm x 1.6mm
  - CPU Module: 72mm x 50mm x 1.6mm
- Debugging Port: 20-pin, 2.54 mm standard JTAG Port
- PCB Layer:
  - Base board: 4 Layer, Immersion-Gold, Separate Layer, The Ground Signal is lead-free
  - Core board: 6 Layer, Immersion-Gold, Separate Layer, The Ground Signal is lead-free
- Core board Port type: Double row 140 pin Port, 2.0mm pin space
- System Power Consumption:
  - Base board & Core board: about 12V/0.16A
  - Base board & Core board & 4.3-Inch Screen: about 12V/0.22A
  - Base board & Core board & 7.0-Inch Screen: about 12V/0.45A

Processor

- AT91SAM9G15/G25/G35/X25/X35 (32 bits ARM RISC processor) runs at up to 400MHz
- 16KB Data Cache, 16KB Instruction Cache

Memory

- 32KB Chip SRAM, 64KB Chip ROM
- 256MB NandFlash, 4MB DataFlash
- 128MB DDR2 SDRAM.

Audio and Video Port

- A 3.5mm Audio Input Port
- A Two-channel 3.5mm Audio Output Port

LCD Touch-Screen Port

- 18 bits True Color
- Resolution: 480x272 for 4.3-Inch Screen and 800x480 for 7.0-Inch Screen, the highest resolution can reach up to 800 x 600

Note: Only MYD-SAM9G15-V2, MYD-SAM9G35-V2 and MYD-SAM9X35-V2 have graphical output function.

Transmission Port

- Standard JTAG Port
- Micro SD Card Port
- Serial Ports
- Four 5-lines UART, two 3-lines UART (The SAM9G15, SAM9G35 and SAM9X35 add the default 232 level; Users can customize LVTTL level.)
- The RS485 and USART0 share the same port, which can be switched by jumper.
- Two CAN Ports (Only MYD-SAM9X25-V2 and MYD-SAM9X35-V2 have CAN Port)
- One High-speed USB THE HOST Port
- One Mini USB THE HOST/DEV Port
- One Full-speed USB THE HOST Port
- One USB Serial Debugging USB_UART Port
- Ethernet MAC
  - MYD-SAM9G15-V2 doesn't have Ethernet MAC
  - MYD-SAM9X25-V2 has two Ethernet MACs (J10 and J11)
  - The others (MYD-SAM9G25-V2\MYD-SAM9G35-V2\MYD-SAM9X35-V2) have only one Ethernet MAC.

LED Indicator

- One Power Indicator (CPU Module: Red)
- One Users Light/System Heartbeat Light (CPU Module: Blue)
- One Power Indicator (Base board: Red)

1.4 Products Packing

No	Name	Number	Note
1	MYD-SAM9X5-V2 Development Board	1	Base board&CPU Module
2	1.5 Meters Crossover Cable	1	
3	1.5 Meters Mini USB 2.0 Cable	2	
4	12V/1.25A DC Power Adapter	1	
5	Product DVD	1	Include Schematic (PDF), User Manual, Source Code, etc.
6	4.3/7.0 Inch LCD Touch Screen	1	Optional

Table 1-1

Chapter 2 Hardware Resource Introduction

2.1 CPU module & Base Board Resources Overview

The features of SAM9x5-V2 series are shown in Table 2-1:

Name		9X25	9X35	9G15	9G25	9G35
Processor		AT91SAM9G15/9G25/9G35/9X25/9X35(ARM926EJ-S Core, frequency at up to 400MHz)				
Memory		128MB DDR2 SDRAM				
Flash		256MB NandFlash; 4MB serial DataFlash				
EEPROM		64KB serial EEPROM				
USB	USB THE HOST	2	2	2	2	2
	USB OTG	1	1	1	1	1
Audio	Audio Input	1	1	1	1	1
	Audio Output	1	1	1	1	1
Network	ETH Port	2	1	0	1	1
Serial	DBGU Serial	1	1	1	1	1
	UART and USART	6	5	5	6	5
JTAG	JTAG Port	1	1	1	1	1
LCD	Support 4.3/7.0 Inch Touch Screen	0	1	1	0	1
RTC	Support the Battery Backup Time	1	1	1	1	1
Extended Port	20 Pins User Extended Port	2	2	2	2	2
Power	12V Power Input	1	1	1	1	1
SD Card	Micro SD	1	1	1	1	1
CAN	CAN Port	1	1	0	0	0
RS485	RS485 Port	1	1	1	1	1
Button	User Button	2	1	1	2	1
	System Button	2	2	2	2	2

Table 2-1

2.2 CPU Module Introduction

2.2.1 CPU Module Preview

(1) CPU Module

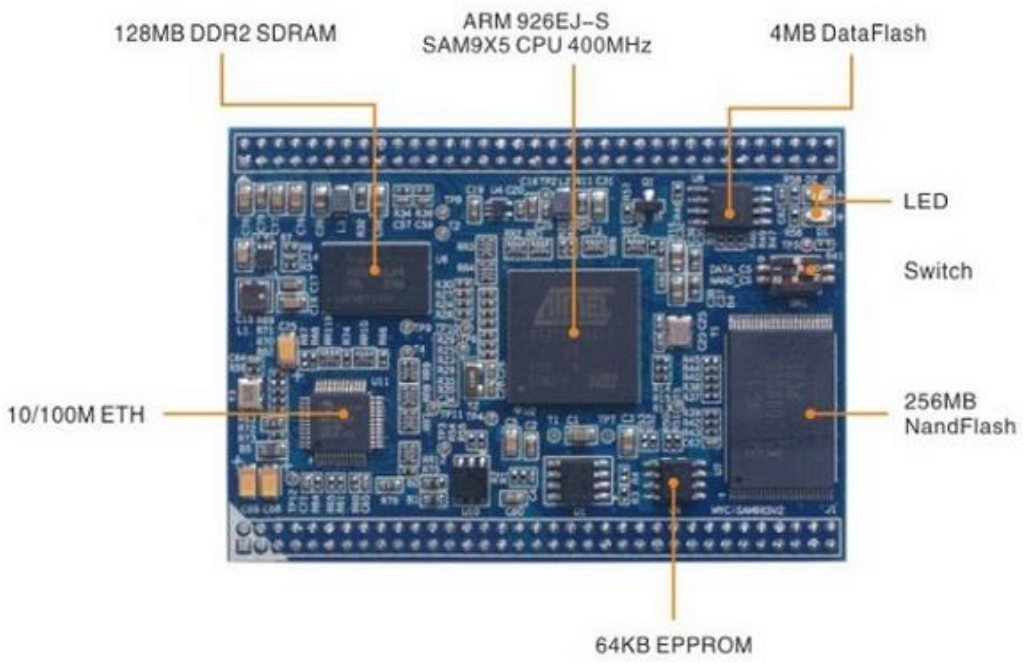


Figure 2-1

(2) Mechanical Dimensions of CPU Module Board

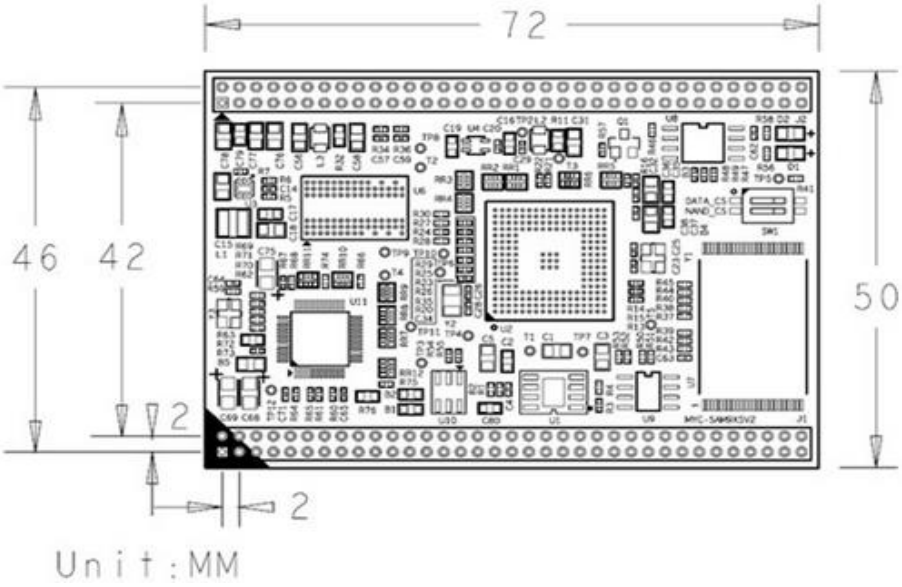


Figure 2-2

2.2.2 CPU Performance

The ARM926EJ-S processor features a technology enhanced 32-bit RISC CPU, flexible size instructions and data caches, Tightly Coupled Memory (TCM) Ports and Memory Management Unit (MMU). It also provides separate instruction and data AMBA AHB Ports for Multi-layer AHB based systems. The ARM926EJ-S processor implements the ARMv5TEJ instruction set which includes an enhanced 16x32-bit multiplier capable of single cycle MAC operations and 16-bit fixed point DSP instructions to enhance performance of many signal processing applications as well as supporting Thumb technology.

2.2.3 CPU Module Board Port definition

CPU module board port definition is shown in Table 2-2(J1-PIN) and Table 2-3(J2-PIN):

Function	IO PIN	PIN	PIN	IO PIN	Function
GND	ETH0-GND	1	2	ETH0-RX+	ETH0-RX+
AVDDT	AVDDT	3	4	ETH0-RX-	ETH0-RX-
ETH0-LED2	ETH0-LED2	5	6	ETH0-TX+	ETH0-TX+
ETH0-LED1	ETH0-LED1	7	8	ETH0-TX-	ETH0-TX-
ETH0-LED0	ETH0-LED0	9	10	GND	GND
GND	GND	11	12	PB0	ERX0
ERX1	PB1	13	14	PB2	ERXER
ERXDV	PB3	15	16	PB4	ETXCK
EMDIO	PB5	17	18	PB6	EMDC
ETXEN	PB7	19	20	GND	GND
ETX0	PB9	21	22	PB8	Empty Signal
AD0	PB11	23	24	PB10	ETX1
AD2	PB13	25	26	PB12	AD1
GND	GND	27	28	PB14	AD3
GPIO	PB15	29	30	PB16	GPIO
AD6	PB17	31	32	PB18	IRQ
ADVERF	ADVERF	33	34	GND	GND
LCDDATA0	PC0	35	36	PC1	LCDDATA1
LCDDAT2	PC2	37	38	PC3	LCDDAT3
LCDDAT4	PC4	39	40	PC5	LCDDAT5
LCDDAT6	PC6	41	42	PC7	LCDDAT7
GND	GND	43	44	PC8	UTXD0

Function	IO PIN	PIN	PIN	IO PIN	Function
UTXD0	PC9	45	46	PC10	LCDDAT10
LCDDAT11	PC11	47	48	PC12	LCDDAT12
LCDDAT13	PC13	49	50	PC14	LCDDAT14
LCDDAT15	PC15	51	52	GND	GND
UTXD1	PC16	53	54	PC17	UTXD1
LCDDAT18	PC18	55	56	PC19	LCDDAT19
LCDDAT20	PC20	57	58	PC21	LCDDAT21
LCDDAT22	PC22	59	60	PC23	LCDDAT23
GND	GND	61	62	PC24	LCDDISP
LCDVSYNC	PC27	63	64	PC25	CTS3
LCDHSYNC	PC28	65	66	PC26	LCDPWM
LCDDEN	PC29	67	68	PC31	FIQ
LCDPCK	PC30	69	70	GND	GND

Table 2-2(J1-PIN)

Function	IO PIN	PIN	PIN	IO PIN	Function
3V3 Power Supply	VDDANA	1	2	VDDBU	3V3 Power Supply
3V3 Power Supply	+3V3	3	4	GND	GND
3V3 Power Supply	+3V3	5	6	GND	GND
3V3 Power Supply	+3V3	7	8	PD20	GPIO
USB-ENA	PD18	9	10	PD19	GPIO
GPIO	PD17	11	12	SHDN	SHDN
LCD-REST	PD16	13	14	WKUP	WKUP
SD-CD	PD15	15	16	NRST	NRST
GND	GND	17	18	NTRST	NRST
USB-C	HHSDPC	19	20	TCK	TCK
USB-C	HHSDMC	21	22	TDO	TDO
GND	GND	23	24	TDI	TDI
USB-B	HHSDMB	25	26	TMS	TMS
USB-B	HHSDPB	27	28	RTCK	RTCK
GND	GND	29	30	PA10	DTXD
USB-OTG	HHSDMA	31	32	PA9	DRXD
USB-OTG	HHSDPA	33	34	PA15	MCI0_DA0
GND	GND	35	36	PA18	MCI0_DA1
Empty Signal	NC	37	38	PA19	MCI0_DA2
Empty Signal	NC	39	40	PA20	MCI0_DA3
GND	GND	41	42	PA17	MCI0_CK
RF	PA29	43	44	PA16	MCI0_CDA
RK	PA28	45	46	PA30	TWD0
RD	PA27	47	48	GND	GND
TD	PA26	49	50	PA31	TWCK0
TF	PA25	51	52	PA23	SPI1-SOCK

Function	IO PIN	PIN	PIN	IO PIN	Function
TK	PA24	53	54	PA22	SPI1-MOSI
CANTX1	PA5	55	56	PA21	SPI1-MISO
CANRX1	PA6	57	58	PA8	RXD2
GND	GND	59	60	PA7	TXD2
SPI0-SPCK	PA13	61	62	PA14	SPI0-NPCS0
SPI0-MOSI	PA12	63	64	PA4	SCK0
SPI0-MISO	PA11	65	66	PA2	RTS0
CTS0	PA3	67	68	PA0	TXD0
RXD0	PA1	69	70	GND	GND

Table 2-3(J2-PIN)

2.2.4 LED

The System LED and User LED circuits are shown in Figure 2-3:

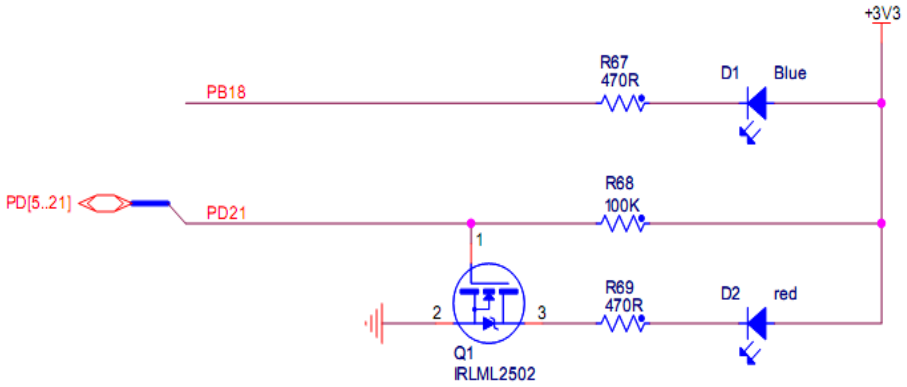


Figure 2-3

2.2.5 Switch Setting

The switch is shown in Figure 2-4:

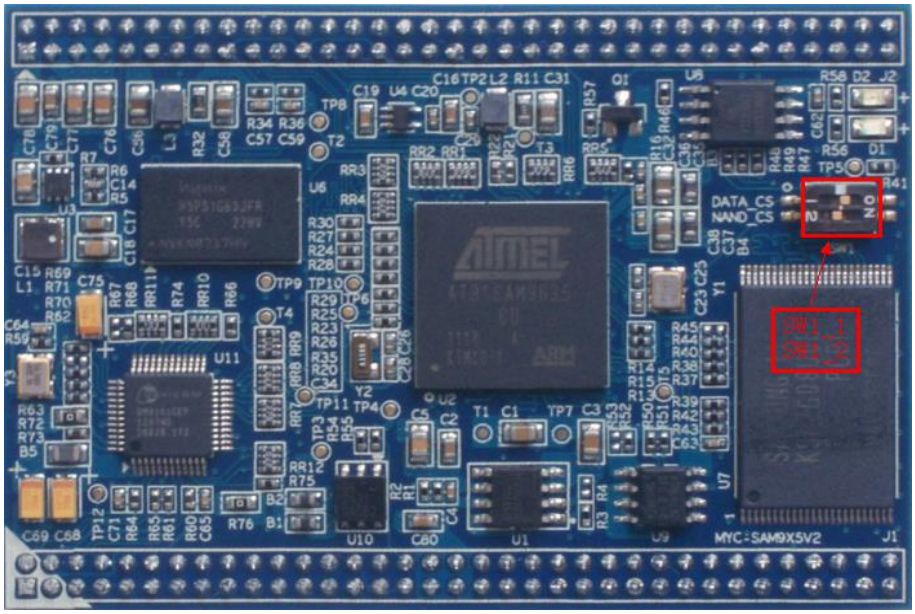


Figure 2-4

The switch functions are shown in Table 2-4:

NO	Function Description	
	ON	OFF
SW1_1	Enable NandFlash	Disable NandFlash
SW1_2	Enable DataFlash	Disable DataFlash

Table 2-4

2.3 Base Board Introduction

2.3.1 Base Board Dimensions

The Base Board Mechanical Dimensions are shown in Figure 2-5:

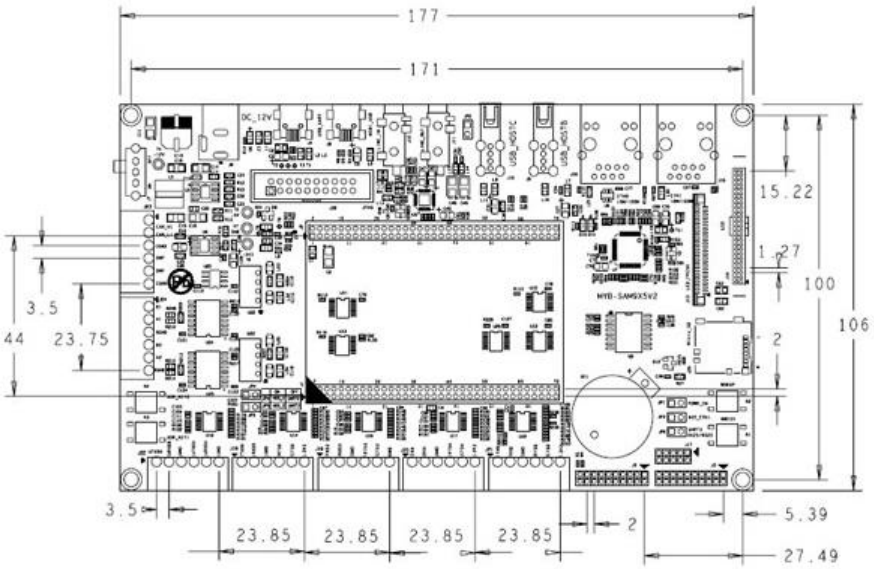


Figure 2-5

2.3.2 Universal Serials

There are two 3-lines UART and two 5-lines USART on the base board as shown in Figure 2-6 to Figure 2-8:

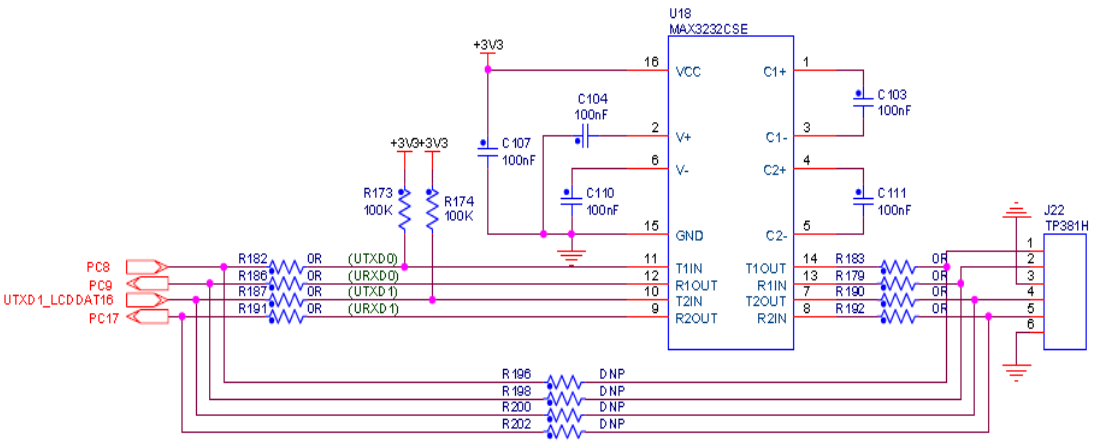


Figure 2-6

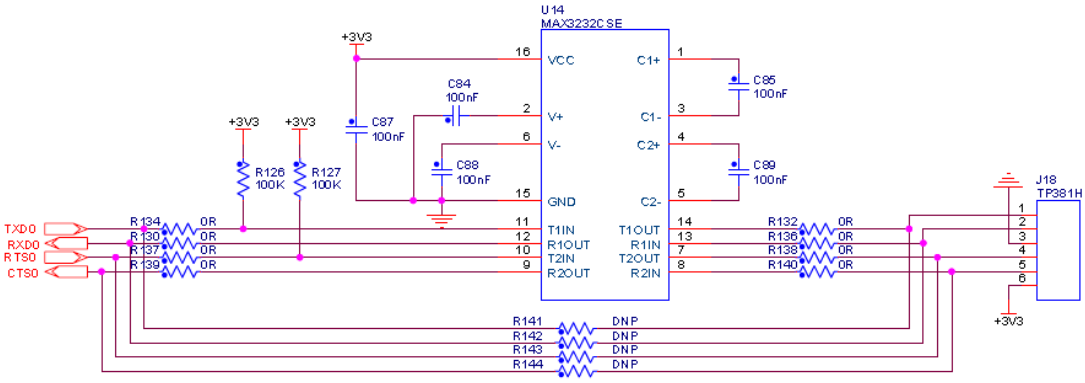


Figure 2-7

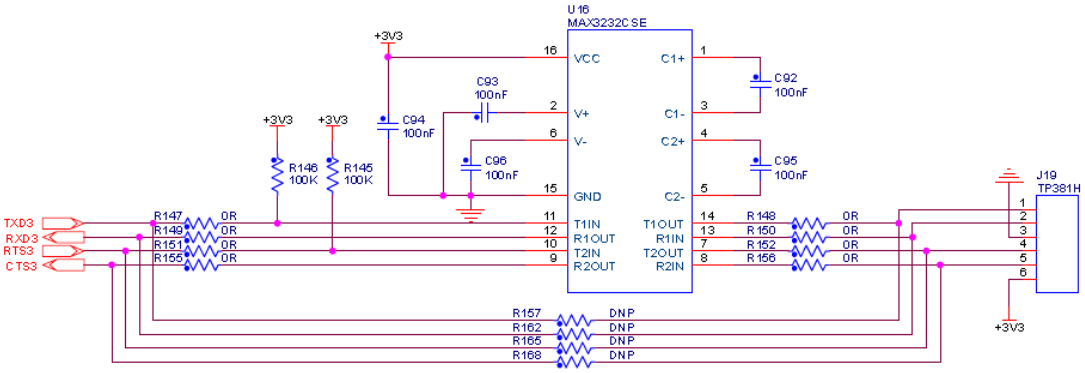


Figure 2-8

2.3.3 CAN BUS

SAM9X25 and SAM9X35 have one CAN Port by TJA1050 chip. Its key features are as follows:

- Fully compatible with the “ISO 11898” standard
- High-speed (up to 1Mbit/s)
- Very Low Electro-Magnetic Emission (EME)
- Different receiver with wide common-mode range for high Electro-Magnetic Immunity (EMI)
- Unpowered node does not disturb the bus lines
- Transmit Data (TXD) dominant time-out function
- Silent mode in which the transmitter is disabled
- Bus Pins protected against transients in an automotive environment
- Input levels compatible with 3.3V and 5V devices

- Output device protected by temperature
- Short-circuit proof to battery and to ground
- At least 110 nodes can be connected

The CAN Bus circuit is shown in Figure 2-9:

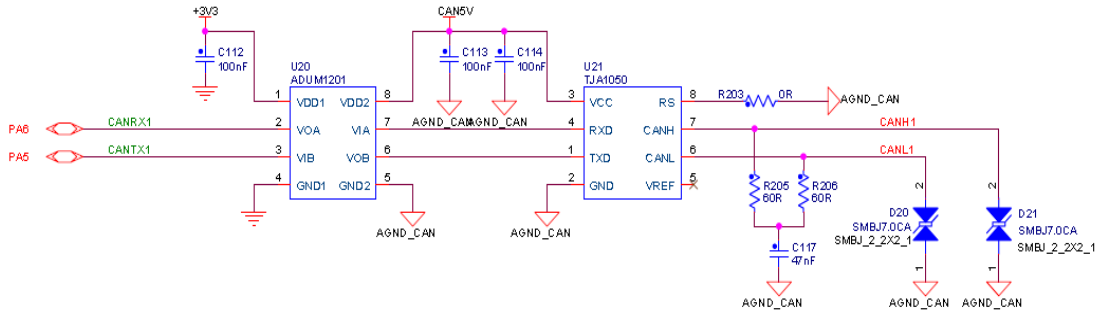


Figure 2-9

2.3.4 JTAG Port

The JTAG Port which has 20 pins is shown in Figure 2-10:

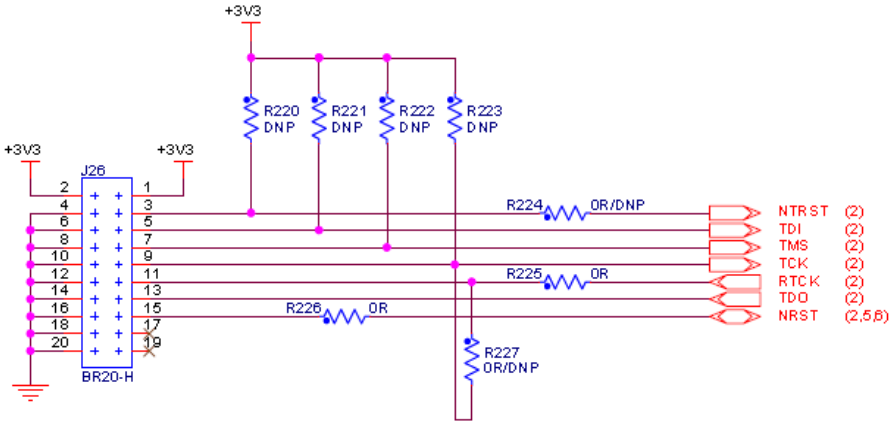


Figure 2-10

2.3.5 LCD Port

The LCD Hardware circuit is shown in Figure 2-11:

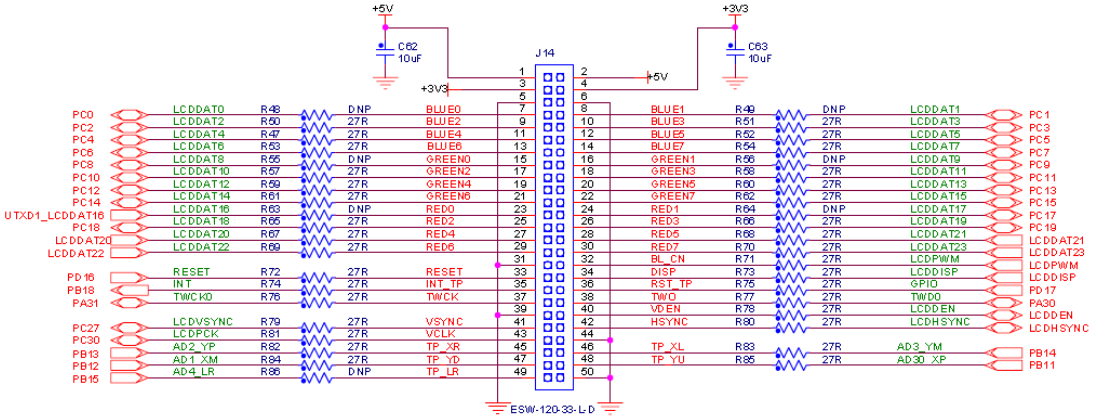


Figure 2-11

2.3.6 User Ports

The User Circuit is shown in Figure 2-12:

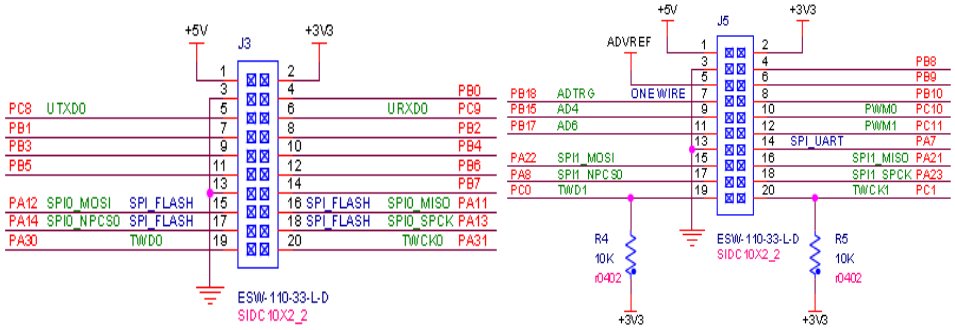


Figure 2-12

2.3.7 Audio Module WM8731

- Audio Performance
  - 97dB SNR (‘A’ weighted @ 48kHz) ADC
  - 100dB SNR (‘A’ weighted @ 48kHz) DAC
  - 1.42–3.6V Digital Supply Operation
  - 2.7–3.6V Analogue Supply Operation
- ADC and DAC Sampling Frequency: 8kHz–96kHz
- Selectable ADC High Pass Filter
- 2 or 3-Wire MPU Serial Control Port

- Programmable Audio Data Port Modes
  - I^2S, Left, Right Justified or DSP
  - 16/20/24/32 Bits Word Lengths
  - Master or Slave Clocking Mode
- Stereo Sound Output and Input
- The Output and Input Volume Control
- Highly Efficient Headphone Driver
- Playback only 18mW
- Analog Pass Through Power only 9mW
- Available in 28-lead SSOP or 28-lead QFN package

The WM8731 circuit is shown in Figure 2-13:

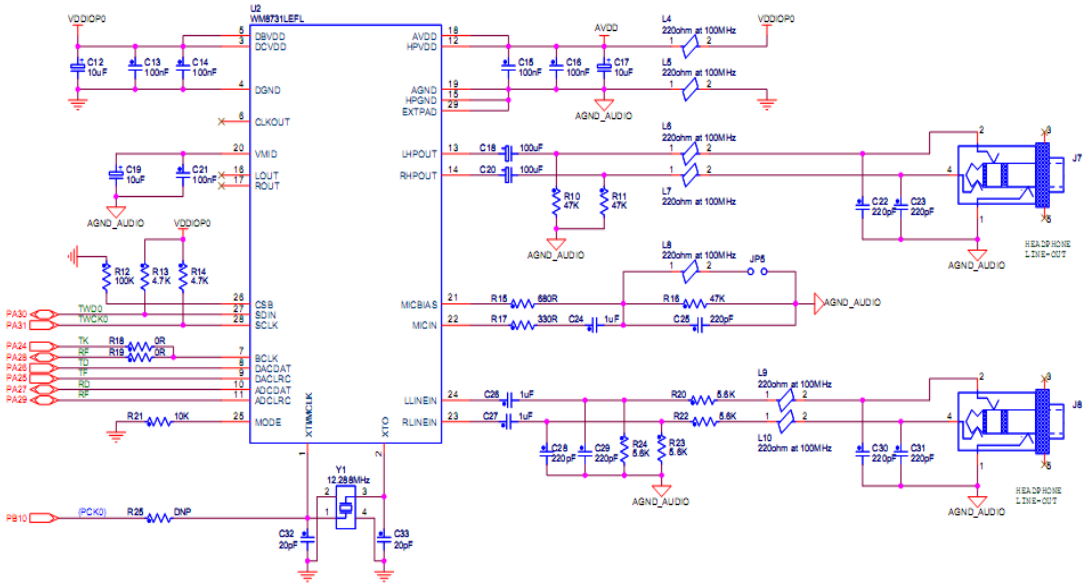


Figure 2-13

2.3.8 USB Module

(1) Mini USB THE HOST/Device chooses AIC1526. Its key features are as follows:

- 110mΩ (5V Input) High-side MOSFE Switch
- 500mA Continuous Load Current per Channel
- 110μA Typical On-state Supply Current
- 1μA Typical Off-state Supply Current
- Current-limit/short Circuit Protection

- Thermal Shutdown Protection under Overcurrent Condition
- Under Voltage Lockout Ensures that Switch is off at Start Up
- Output can be Forced Higher than Input(Off-State)
- Open-Drain Fault Flag
- Slow Turn ON and Fast Turn OFF

The USB THE HOST/Device Circuit is shown in Figure 2-14:

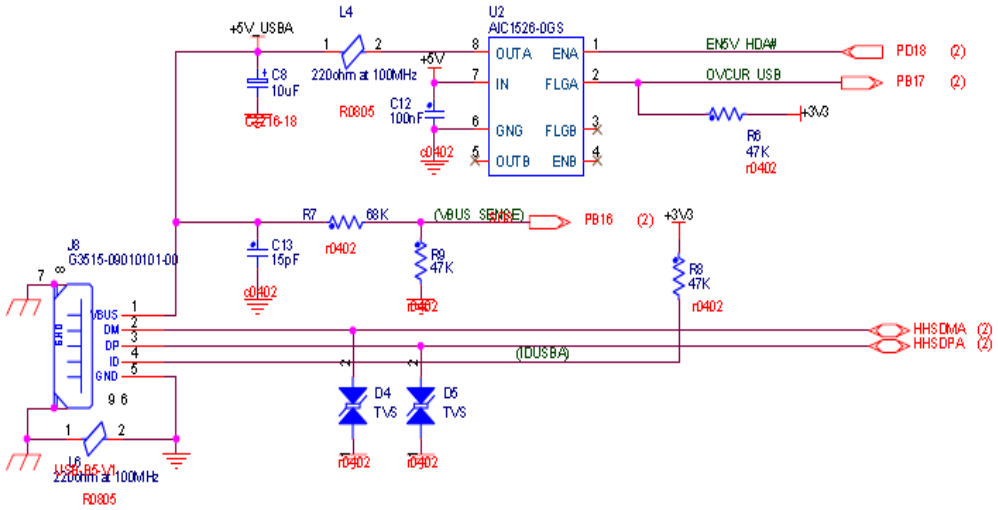


Figure 2-14

(2) The USB THE HOST Circuit is shown in Figure 2-15:

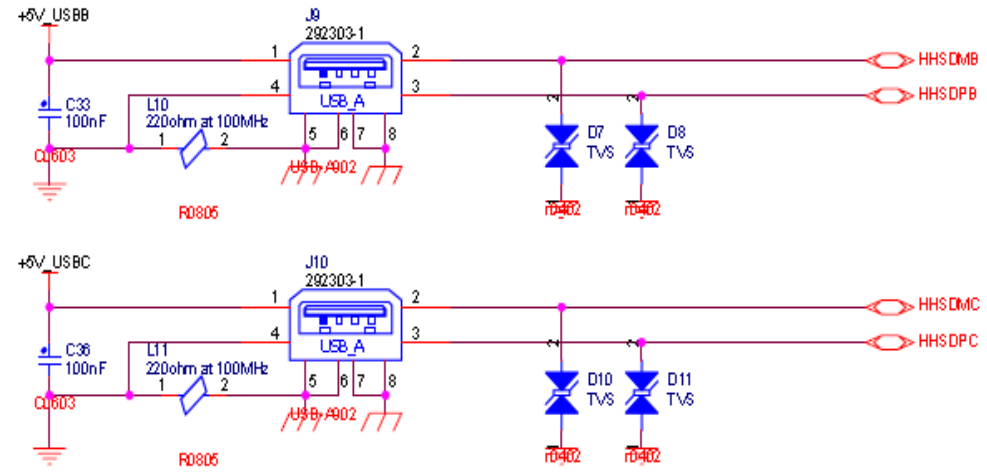


Figure 2-15

(3) Mini USB DBUG module chooses CP2102, Its key features are as follows:

- Embedded USB transceiver, without external circuit
- It contains clock circuit, without an external circuit
- It contains reset circuit on the electricity

- Output Voltage Regulation can be within 3.3 V voltage
- Conformance to the specification of USB2.0
- SUSPEND pin support USB state hung
- Data Formats Supported:

  Data bits: 5, 6, 7, and 8

  Stop bits: 1, 1.5, and 2

  Parity: odd, even, mark, space, no parity
- Integrate a 512 bytes receiving buffer and a 512 bytes sending buffer
- Support hardware or X - ON/X - OFF shake hands

The Mini USB DBUG circuit is shown in Figure 2-16:

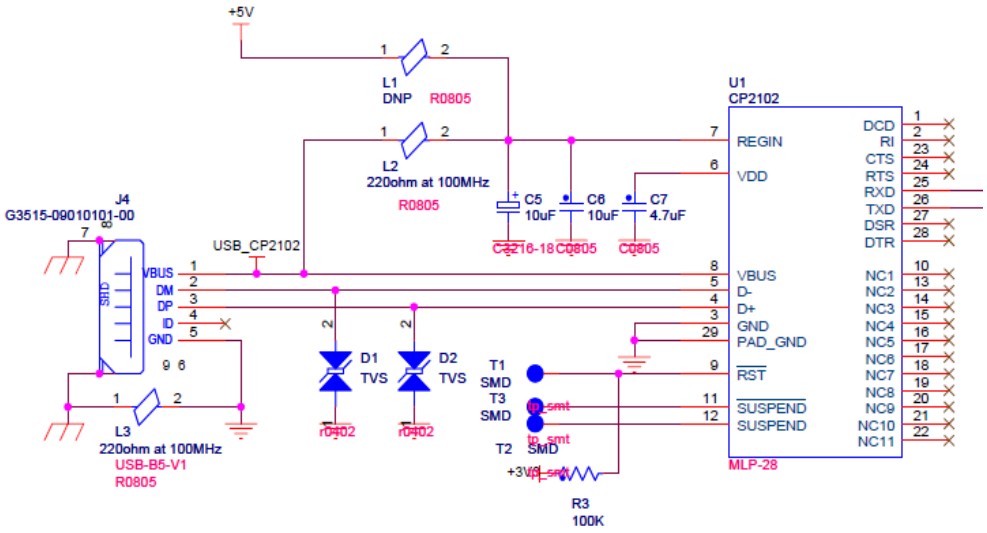


Figure 2-16

2.3.9 RS485 Bus

SAM9G25 and SAM9X25 have two 485 Ports, The other three 9X5 boards have one RS485 Port.(signal reuse with serial Port, with electric isolation). RS485 Bus uses ADM2483 chip and its key features are as follows:

- In line with the ANSI TIA/EIA RS-485-A-1998 and ISO 1998:1987 (E) standard
- Data Rate: 250 KBPS
- Limit Set Rate Driver Output
- Low Power Consumption: 4.5 mA (maximum)

- Working Voltage (VDD1) : 5 V or 3 V
- High Common Mode Transient Immunity: > 25 KV/(including s
- Burr on the Electric on/off Protection
- Thermal-shutdown Protection
- Receiver Open-circuit Fault Secure Design

The RS485 circuit is shown in Figure 2-17:

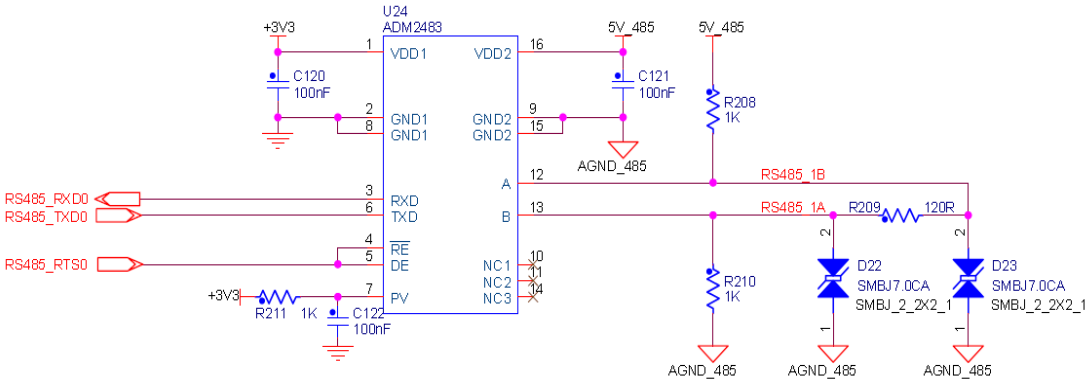


Figure 2-17

2.3.10 External UART

MYD-SAM9X5-V2 extends two 5-lines UART by SC16IS Port. Its key features are as follows:

- Full Duplex UART
- IIC,SPI Port
- Working Voltage:3.3 V
- Industrial Temperature Range: - 40 ℃ to + 95 ℃
- 64 bytes in FIFO (Sending and Receiving)
- Automatically using Hardware Flow Control
- Automated Software Flow Control
- Single, Double X-on/X-off Characters
- Eight Programmable IO pin
- Support RS485

The External 5-lines UART circuit is shown in Figure 2-18:

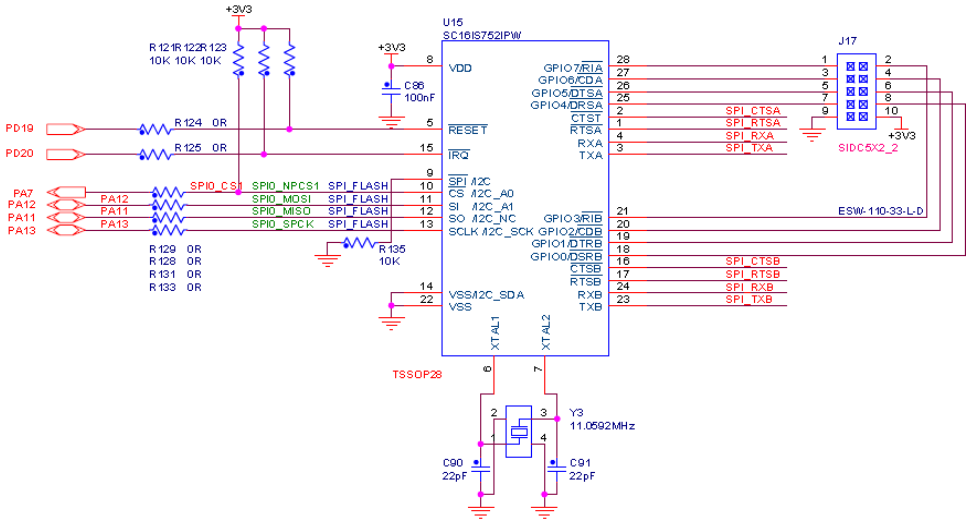


Figure 2-18

2.3.11 External RTC

MYD-SAM9X5-V2 extends RTC by DS3231. Its key features are as follows:

- Backup Battery Power
- Working Temperature
  - Commercial Grade: 0 ℃ ~ + 70 ℃
  - Industrial Grade: - 40 ℃ ~ + 85 ℃
- Low Power Consumption
- Real Time Clock Calculation, points, day, week, month, year
- Time Alarm
- Programmable Square Wave Output
- Working Voltage:3.3 V

The External RTC circuit is as shown in Figure 2-19:

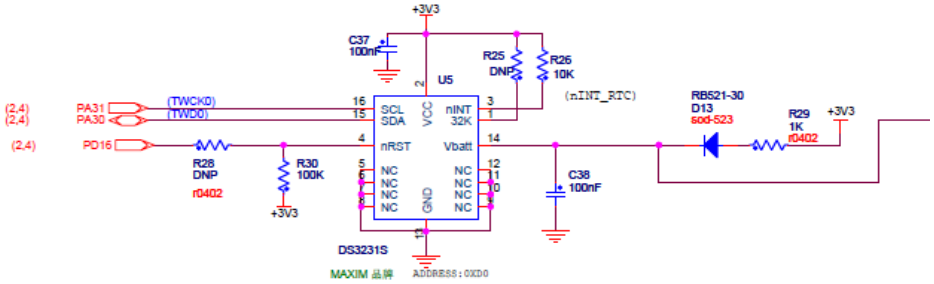


Figure 2-19

2.3.12 ETHERNET

The ETHERNET module uses DM9161AEP chip. Its key features are as follows:

- Support 10/100 MBPS IEEE802.3 / IEEE802.3 U network transport protocol
- Support the MDI/MDI - X Automatic Crossing
- Alternative Full Duplex or Half Duplex Operation
- MII Management Port, Maskable Interrupt Output Capacity
- Provide Circuit Model to Facilitate Diagnosis System
- Support Automatic Consultation Function, Conformance to IEEE802.3 U Network Transport Protocol
- 48-pin LQFN Packaging
- Compatible with 3.3 v and 5.0 v Input and Output

The ETHERNET circuit is shown in Figure 2-20:

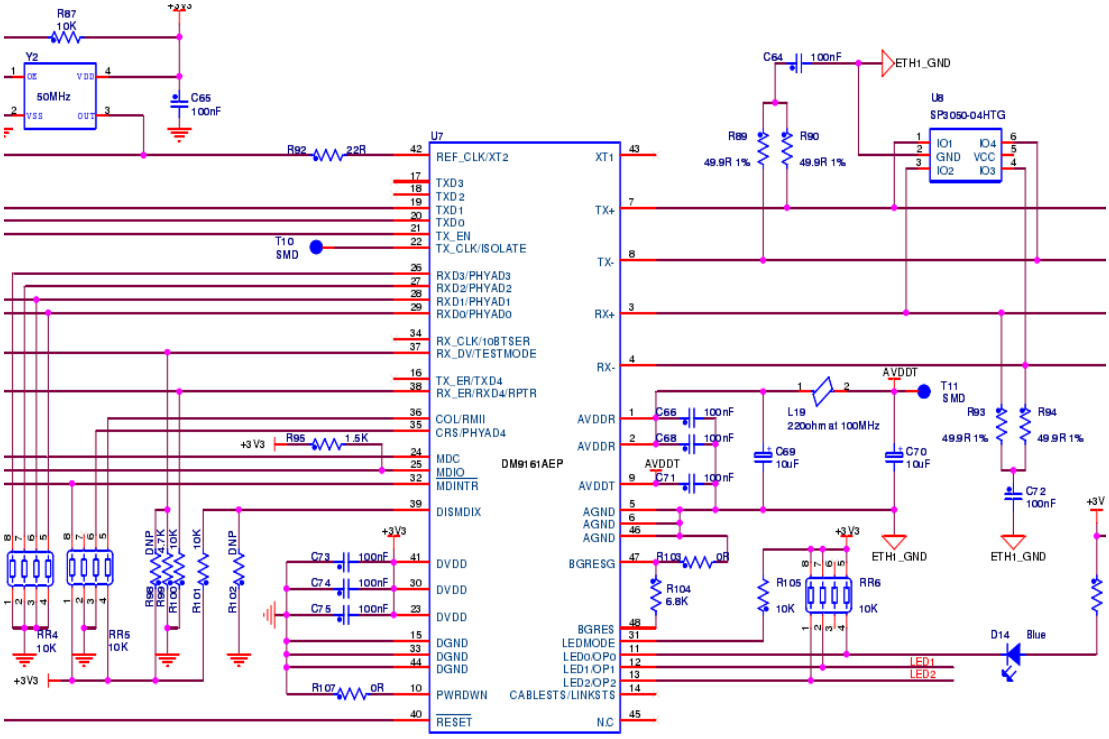


Figure 2-20

2.4 Jumper Settings

(1) JP1, JP3, JP4, JP5 and JP6 are shown in Figure 2-21:

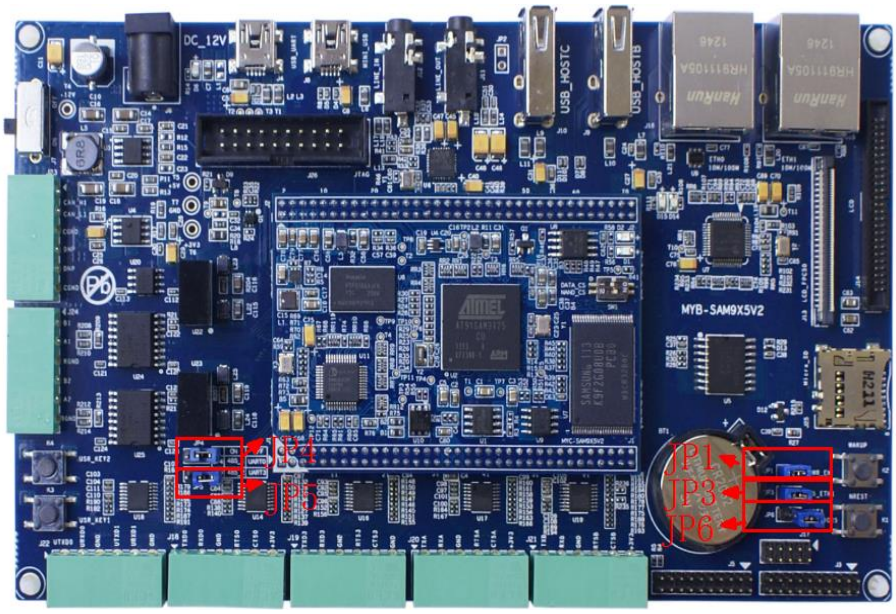


Figure 2-21

(2) The functions of JP1, JP3, JP4, JP5 and JP6 are as shown in Table 2-5:

Jumper	Functional Description	
	CLOSE(Connection)	OPEN(Disconnection)
JP1	Force Power On	Normal Mode
JP3	Enable ETH1,Disconnect LCD	Enable LCD, Disconnect ETH1
JP4	Enable RS485_1, Disconnect USART0	Enable USART0,Disconnect RS485_1
JP5	When JP6 CLOSE, Enable RS485_2, Disconnect USART3. JP6 OPEN invalid	When JP6 CLOSE, Enable USART3, Disconnect RS485_2. JP6 OPEN invalid
JP6	Enable USART3,Disconnect LCD	Enable LCD, Disconnect USART3

Table 2-5

Chapter 3 MDK Example Code

3.1 Overview

MDK examples are programed without operating system and the development kits are MDK-ARM 4.53. This Chapter describes how to use and write test codes, which is listed as follows:

(1) Built and configured MDK development environment;

(2) Download, compile and debug MDK sample programs;

(3) The features introduction, usage and appearance explanations of the test procedures, including board start, DMA, ADC, LCD, Storage System, Ethernet and so on.

3.2 Preparation

(1) Install MDK-ARM (Version 4.53) development kits and the license

(2) Prepare for MYD-SAM9X5-V2 development board kits

MYD-SAM9X5-V2 series development boards include:

- MYD-SAM9G15-V2
- MYD-SAM9G25-V2
- MYD-SAM9G35-V2
- MYD-SAM9X25-V2
- MYD-SAM9X35-V2

3.2.1 Configure And Compile

Let's take sample program "getting-started" for example. Firstly find "04-MDK_Source \01_getting-started\Project" folder and double click project file "getting-started.uvproj", then configure project. The detailed steps are as follows (Note: Generally, Download and compile a program by default setting. If the downloading or compiling process failed, it is

necessary to check and reset the program.):

(1) Select the project and click the right button, then select “Option for target MYD-SAM9X35” or press “Alt + F7”. The Setting window is shown in Figure 3-1:

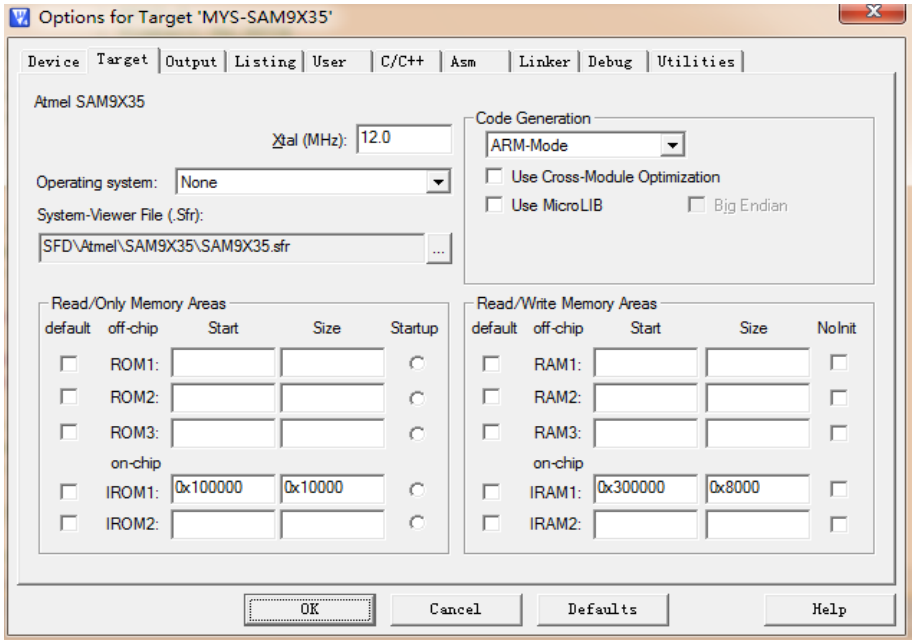


Figure 3-1

(2) Choose “SAM9X35” in tab “Device” as follows: (Note: The specific steps for setting other chips are similar to the example of “SAM9X35”). Steps are shown in Figure 3-2:

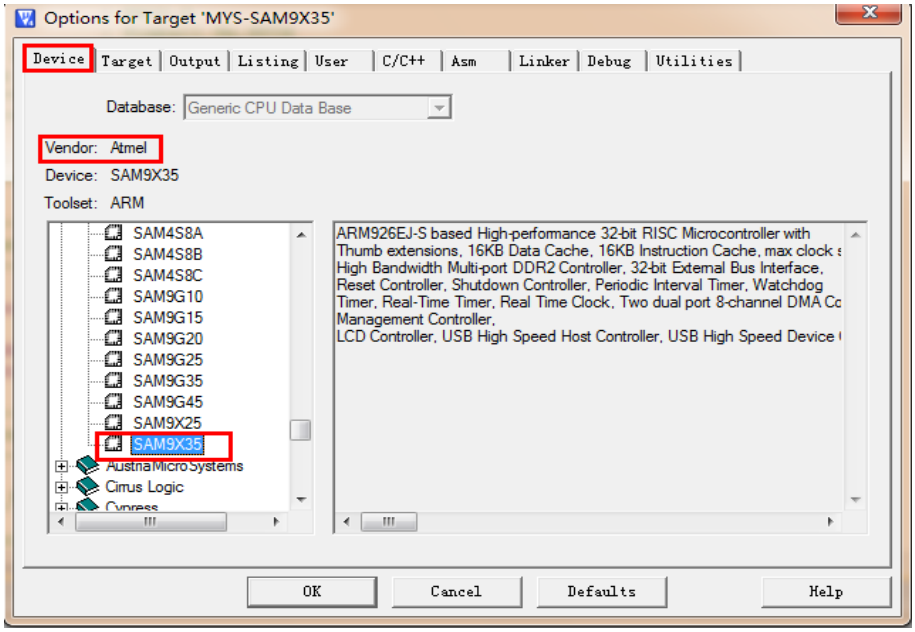


Figure 3-2

When the default Configuration is completed, “Target” tab will have a default

configuration automatically. Refer to Figure 3-1.

(3) “Output” tab (including the intermediate files) is shown in Figure 3-3:

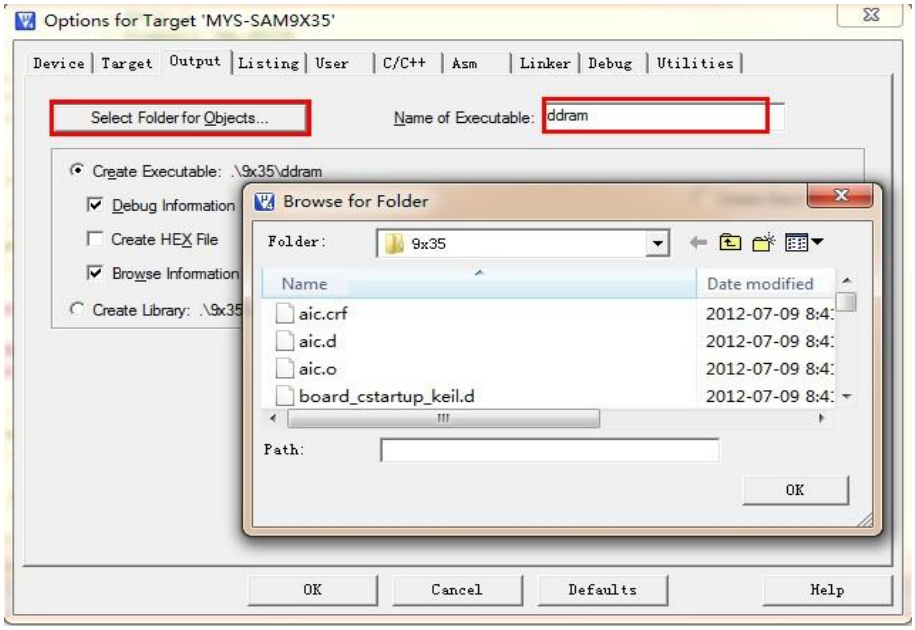


Figure 3-3

Click “Select Folder for Objects…”and then click “OK” on the popup dialog box which we can select the storage path. Users can define the executable file name by themselves.

(4) The generated intermediate file folder can be selected in the “Listing” tab. Refer to Figure 3-4:

Figure 3-4

(5) The “User” tab is shown in Figure 3-5:

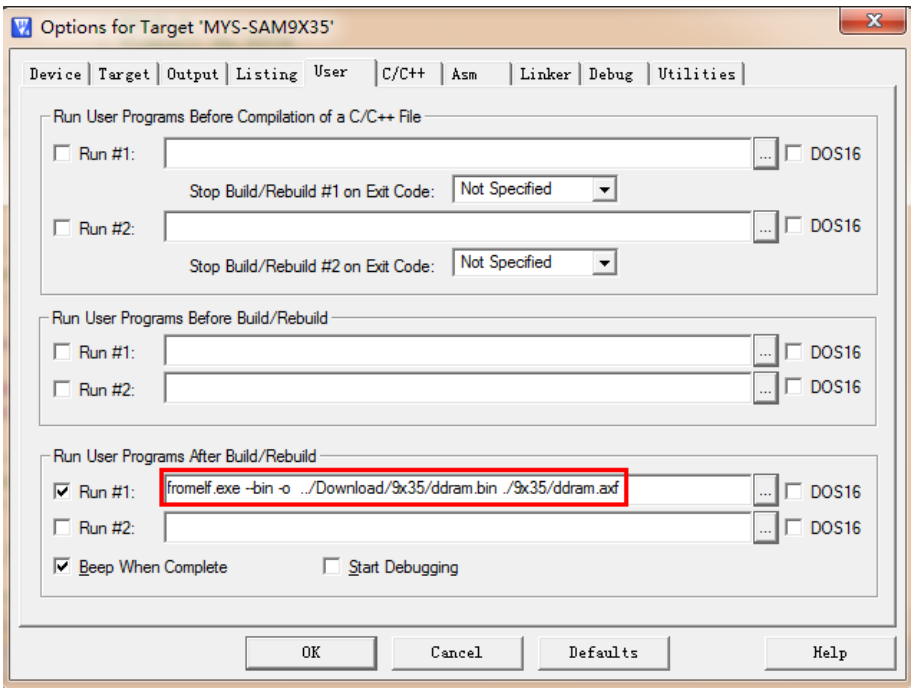


Figure 3-5

The command marked by the red box specifies the storage path of generated executable file and user can modify it.

(6) C/C++ configuration, user can add or delete compile files path. Refer to Figure 3-6:

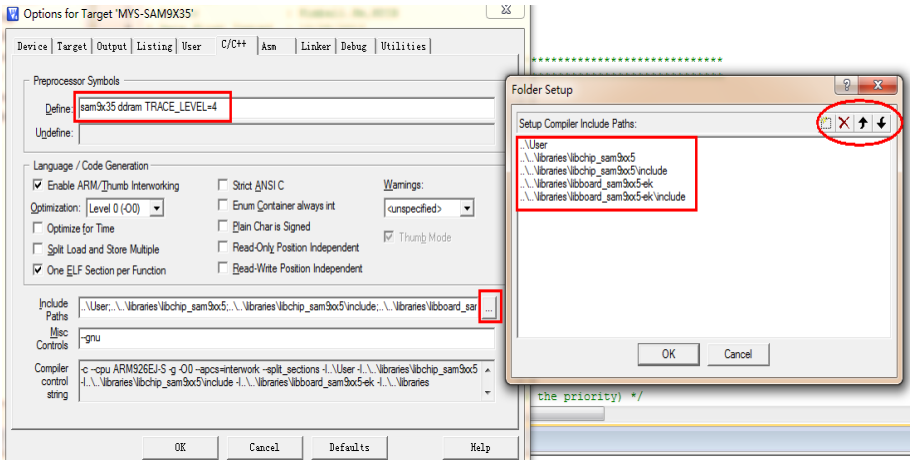


Figure 3-6

(7) The Linker Configuration is shown in Figure 3-7:

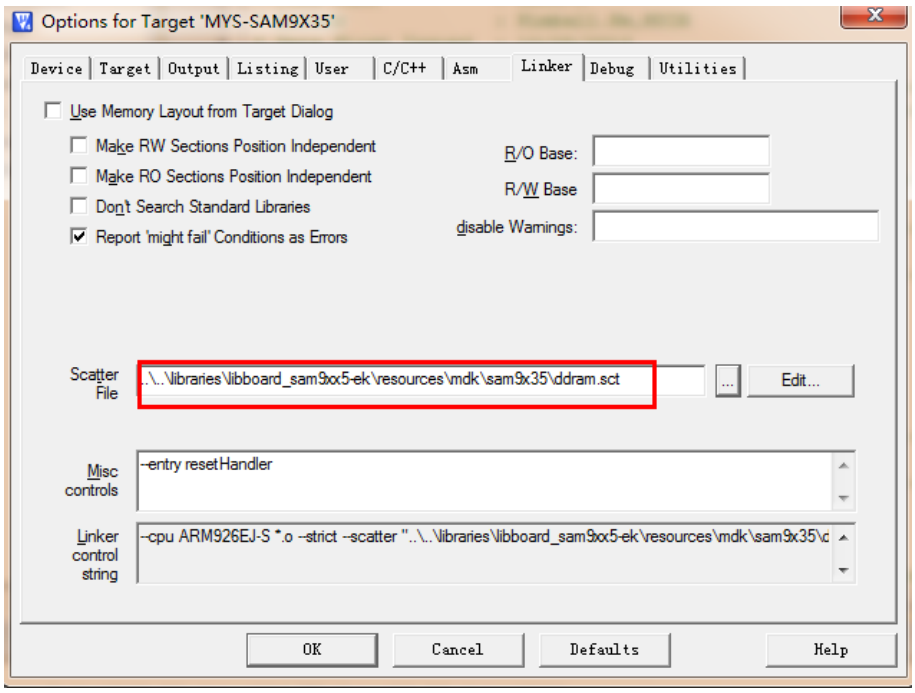


Figure 3-7 (1)

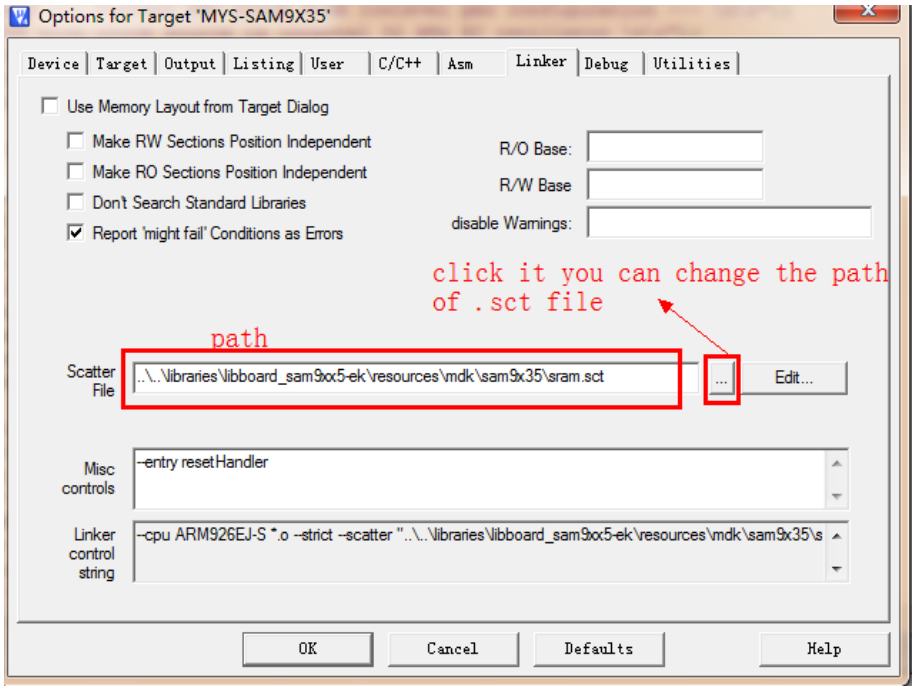


Figure 3-7 (2)

The Linker configuration of “getting-started” project is shown in Figure 3-7(1) (This sample program generates “ddram.bin”, most MDK routines generate “ddram.bin”) and the Linker configuration of “pmc_clock_switching” project is shown in Figure 3-7(2) (generate “sram.bin”). Both select “.sct” file which have different names and must have.

(8) Choose “project->Rebuild all target files” to compile the project, or click on

shortcut icon to compile. The steps are shown in Figure 3-8:

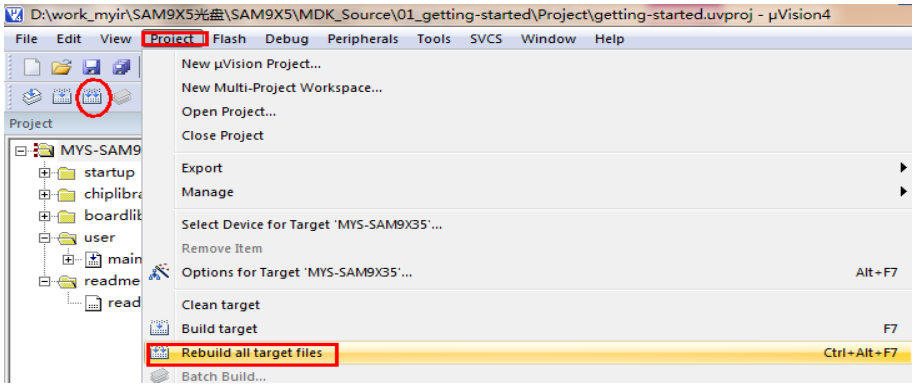


Figure 3-8

There will be a executable bin file in the directory of “Output tab”, There is a prompt about having executed the user command on the “Build Output” dialog. Refer to Figure 3-9:

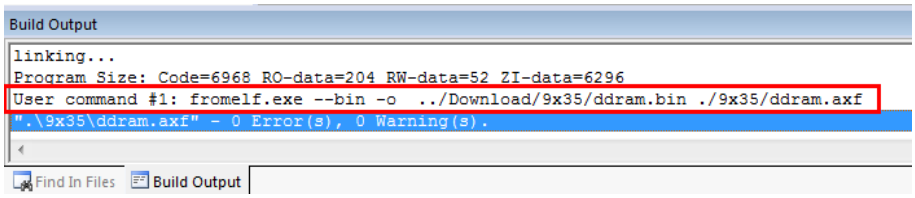

Build Output
linking...
Program Size: Code=6968 RO-data=204 RW-data=52 ZI-data=6296
User command #1: fromelf.exe --bin -o ../Download/9x35/ddram.bin ./9x35/ddram.axf
".\9x35\ddram.axf" - 0 Error(s), 0 Warning(s).
Find In Files
Build Output

Figure 3-9

At this point, the configuration and compilation of MDK routine have been completed.

3.2.2 MDK Routine Debug

The following is the MDK sample program configuration, which needs the hardware emulator “ULink2” in advance (Please contact our company to purchase if needed).

(1) After opening project file, open the setting dialog box and select “Debug”tab. Refer to Figure 3-10:

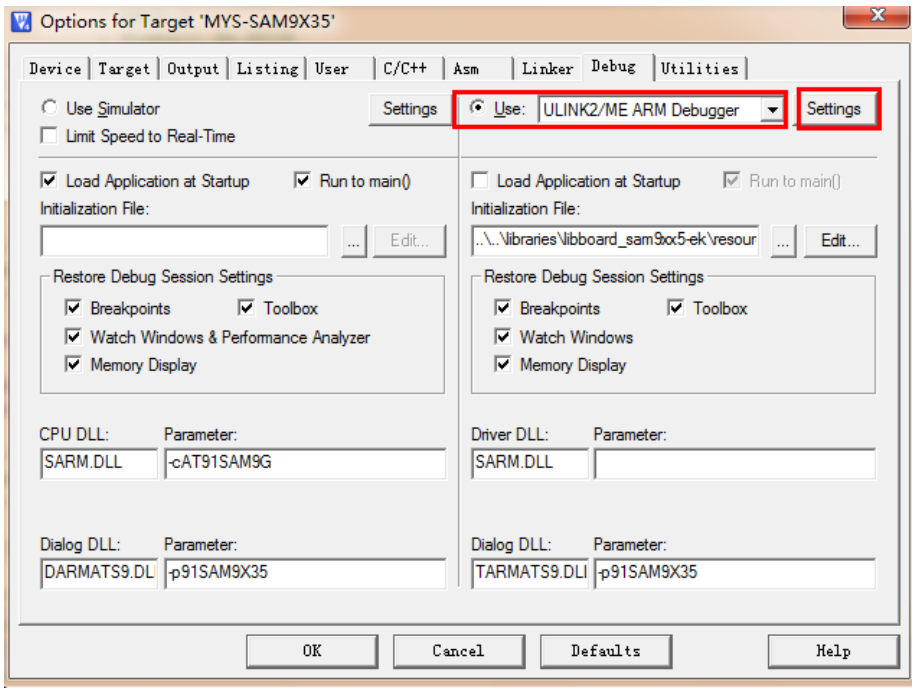


Figure 3-10

(2) Check hardware emulator ULink2

After connecting ULink2 to board, the indicator lights of RUN and COM change blue and then turn off, while the indicator light changes red and remains. Thus, it indicates ULink2 is OK.

(3) Clicking “Settings” tab in Figure 3-10, there will be connection status of ULink2 and the development board as well as kernel identification. Refer to Figure 3-11:

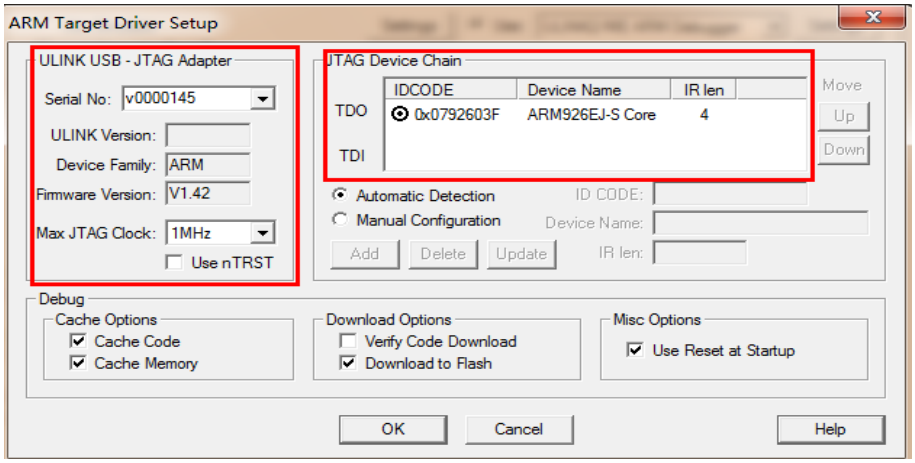


Figure 3-11

(4) Click “Ctrl+F5” or shortcut icon, or select “Debug->Start/Stop Debug Session” to start debug. Refer to Figure 3-12:

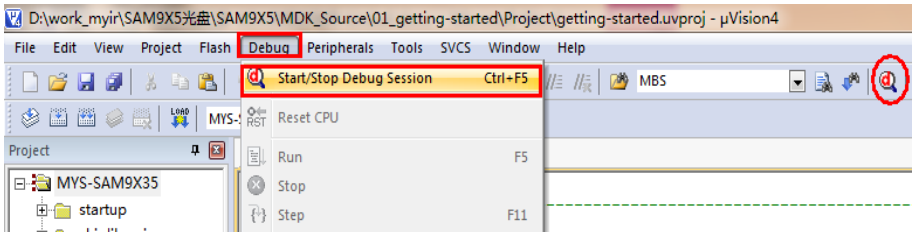


Figure 3-12

3.2.3 Super Terminal Configuration

Super Terminal Configuration

(1) Install USB serial port driver CP210X, run “PreInstaller.Exe”, as shown in Figure 3-13:

Figure 3-13

Choose "install" to install driver program, as shown in Figure 3-14, 3-15:

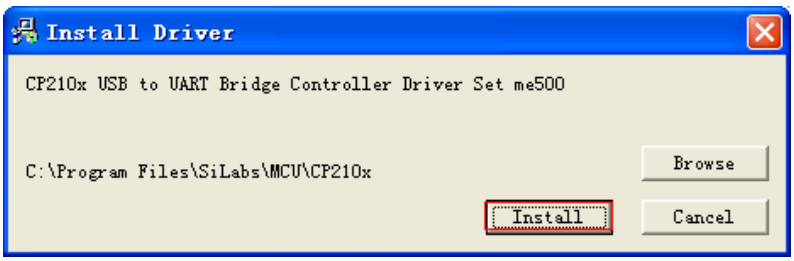


Figure 3-14

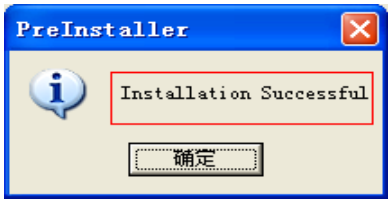


Figure 3-15

(2) Connect USB_UART to PC by mini USB (The first connection needs to install the USB serial port driver). Open the terminal, main parameters must be configured as follows:

Select the corresponding USB_UART port: COMX, such as USB_UART is com1 in device manager, then X is 1; Baud rate: 115200; Data bits: 8; Parity bit: None; Stop bits: 1.

Note: if there are no special statements in MDK sample program tests, DBUG information is output by USB_UART.

Download

(1) Install samba software (Version 2.11 or above, the installation package is in 03-Tools\SAM-BA file folder). If samba (below version 2.11) has been installed, uninstall it cleanly.

(2) Connect board to PC by mini USB and power on.

(3) Turn SW1_1, SW1_2 off, press NRST to reset board, and there will be tips of installing driver after a certain period of time. The device is shown in Figure 3-16:

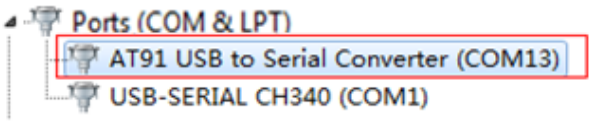


Figure 3-16

Note: Turn SW1_1 on to enable NANDFLASH; turn SW1_2 on to enable DATAFLASH. Turn SW1_1, SW1_2 off to prevent the chip booting from this two medias, thus enabling to connect by USB.

Install driver manually if it has failed in automatic manner. Right click to update driver software program (Note: the sample has been properly installed) and choose to install it manually. Refer to Figure 3-17 to Figure 3-21:

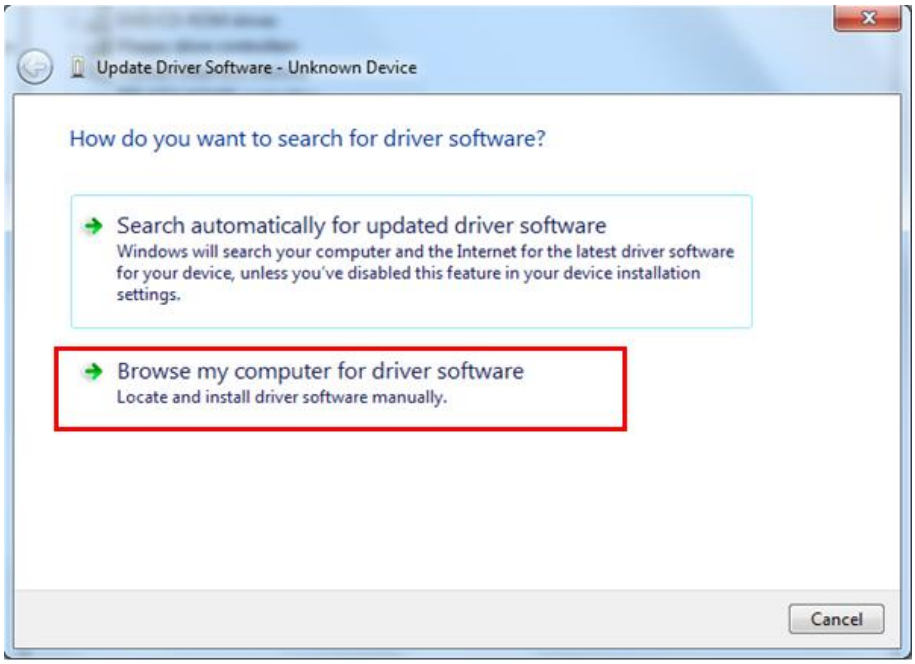

Update Driver Software - Unknown Device
How do you want to search for driver software?
Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.
Browse my computer for driver software
Locate and install driver software manually.
Cancel

Figure 3-17

Choose to find and install driver software manually.

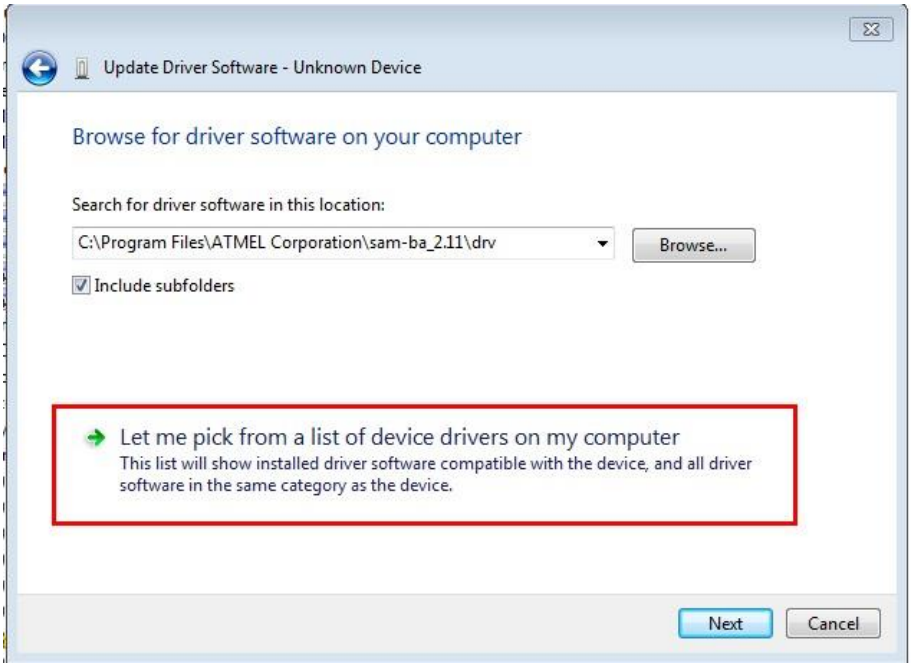

Update Driver Software - Unknown Device
Browse for driver software on your computer
Search for driver software in this location:
C:\Program Files\ATMEL Corporation\sam-ba_2.11\drv
Browse...
Include subfolders
Let me pick from a list of device drivers on my computer
This list will show installed driver software compatible with the device, and all driver software in the same category as the device.
Next
Cancel

Figure 3-18

Click the red box to enter driver list. Refer to Figure 3-19:

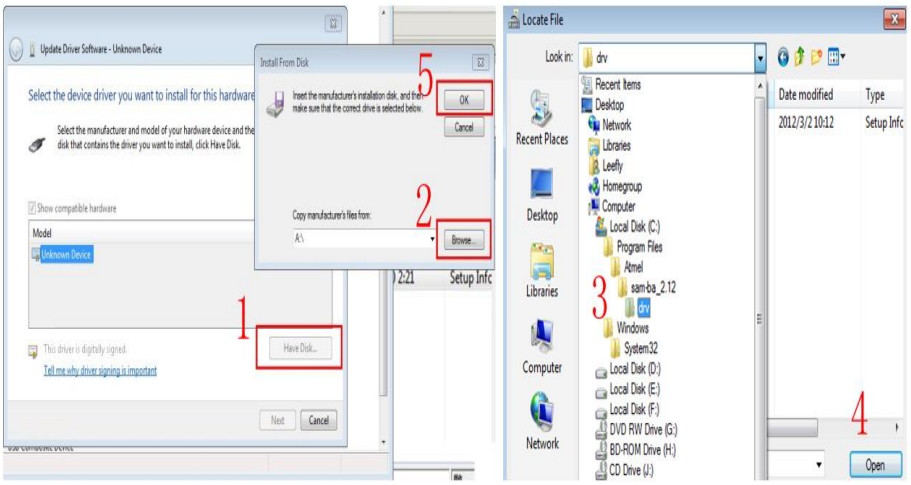


Figure 3-19

Click “OK” to install driver. Refer to Figure 3-20. When installation is finished, it will be shown as Figure 3-21:

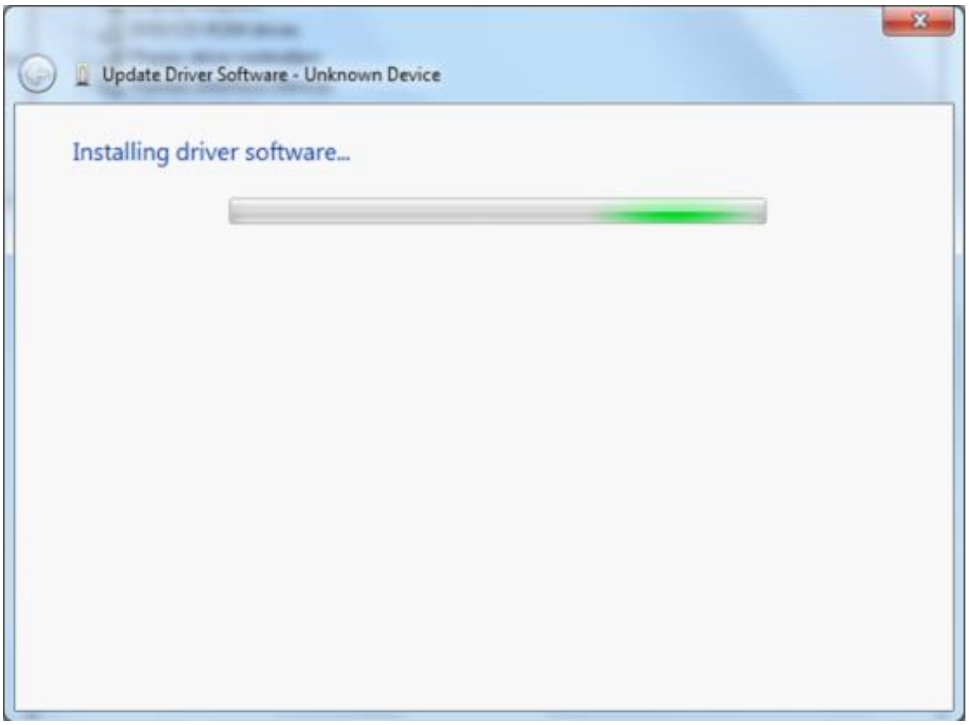


Figure 3-20

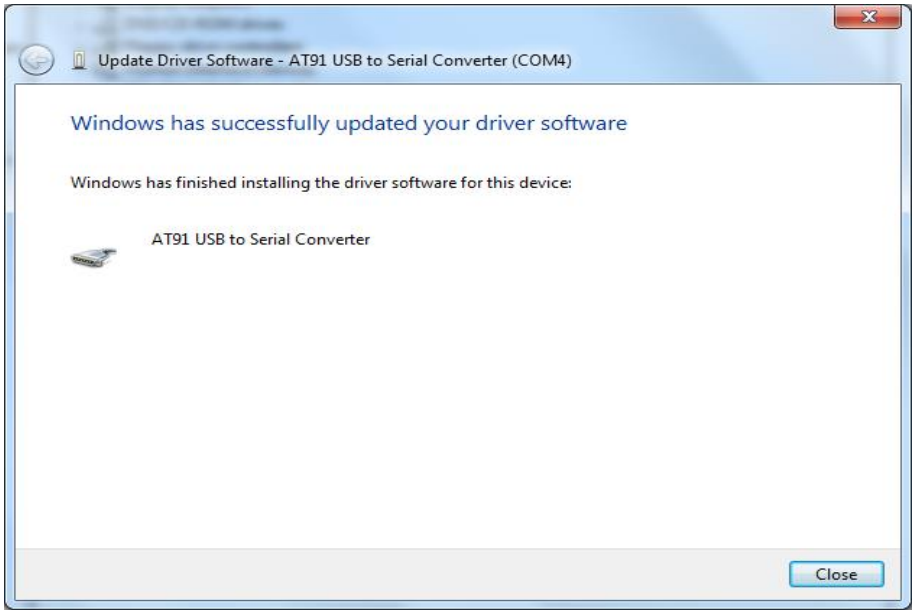

Update Driver Software - AT91 USB to Serial Converter (COM4)
Windows has successfully updated your driver software
Windows has finished installing the driver software for this device:
AT91 USB to Serial Converter
Close

Figure 3-21

After installing samba driver, download program into board. There are two ways: download automatically and manually. Please refer to follow contents for more details.

3.2.4 Manual Download

Let's take downloading "getting-started" to exemplify the process of manual download.

Firstly turn on SW1_1, SW1_2 (Note: disconnect the base board JP8 jump line, otherwise the computer will not recognize the development board) and press NRST to reset board. Then open samba software, the startup port is shown in Figure 3-22:

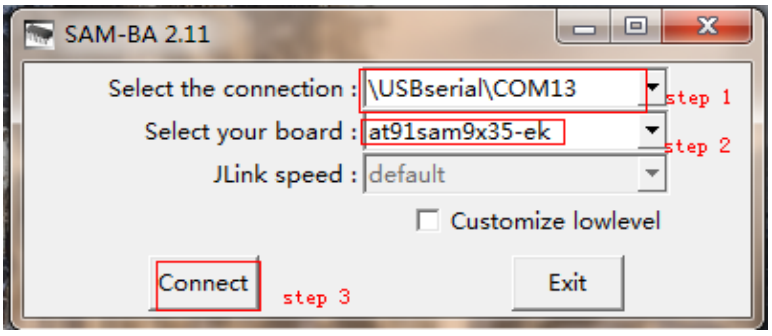


Figure 3-22

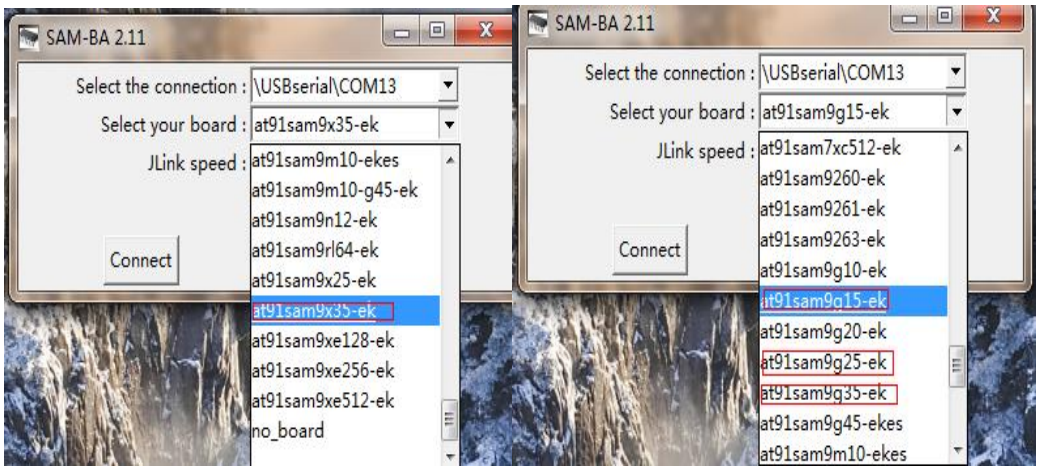


Figure 3-23

Refer to Figure 3-22, the operation is mainly to choose the right board model (Note: this example use MYD-SAM9X35 board, so we select at91sam9x35-ek. If choosing other MYD-SAM9X5-V2 series, it needs to select the corresponding model. Refer to Figure 3-23). The main port is shown after clicking “Connect” in Figure 3-24:

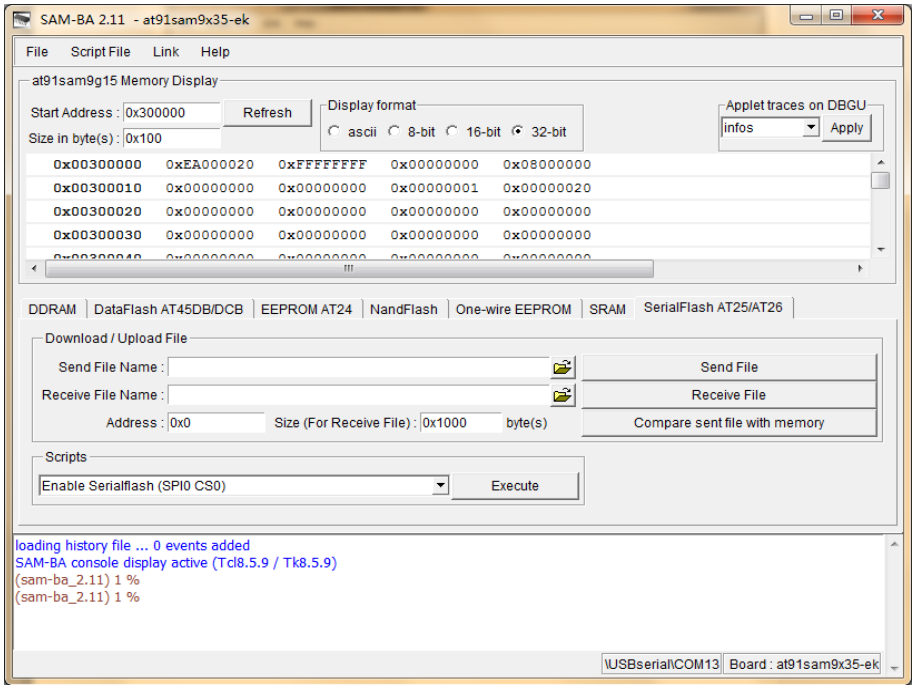


Figure 3-24

After entering the main port, turn SW1_2 on, Refer to Figure 3-25. Firstly choose SerialFlash AT25/AT26, then set the script as Enable SerialFlash, lastly perform “Execute”:

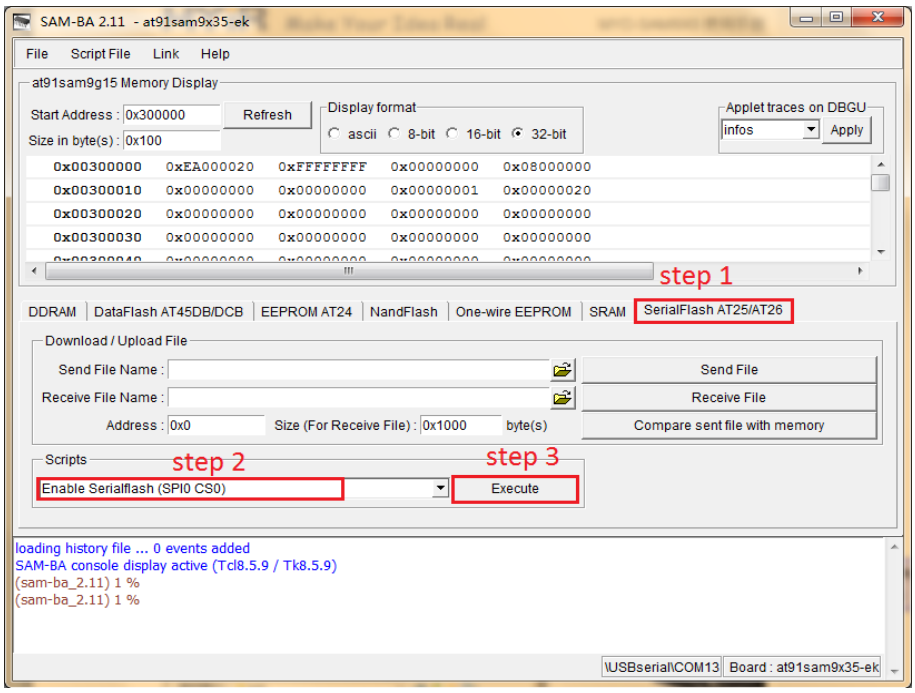


Figure 3-25

After enabling DataFlash, it will show “(sam-ba_2.11) 1 % SERIALFLASH::Init 0” and download “dataflashboot.bin” (Location: “04-MDK_Source\01_getting-started\Download\9x35”). The specific operation is shown in Figure 3-26:

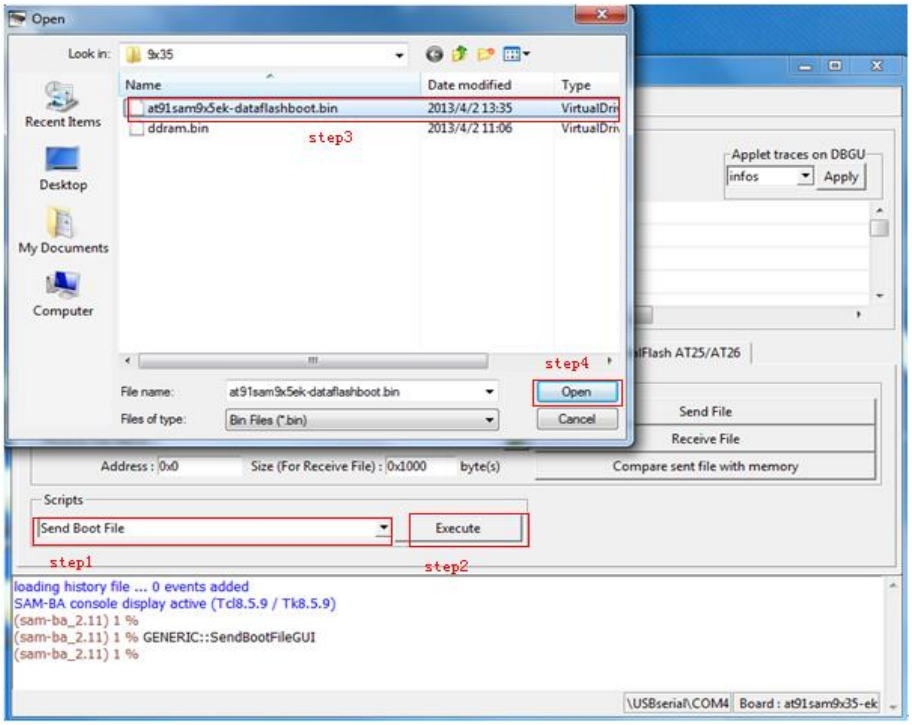


Figure 3-26

Download “ddram.bin” after downloading dataflash boot file, then download

"ddram.bin". The specific operation is shown in Figure 3-27:

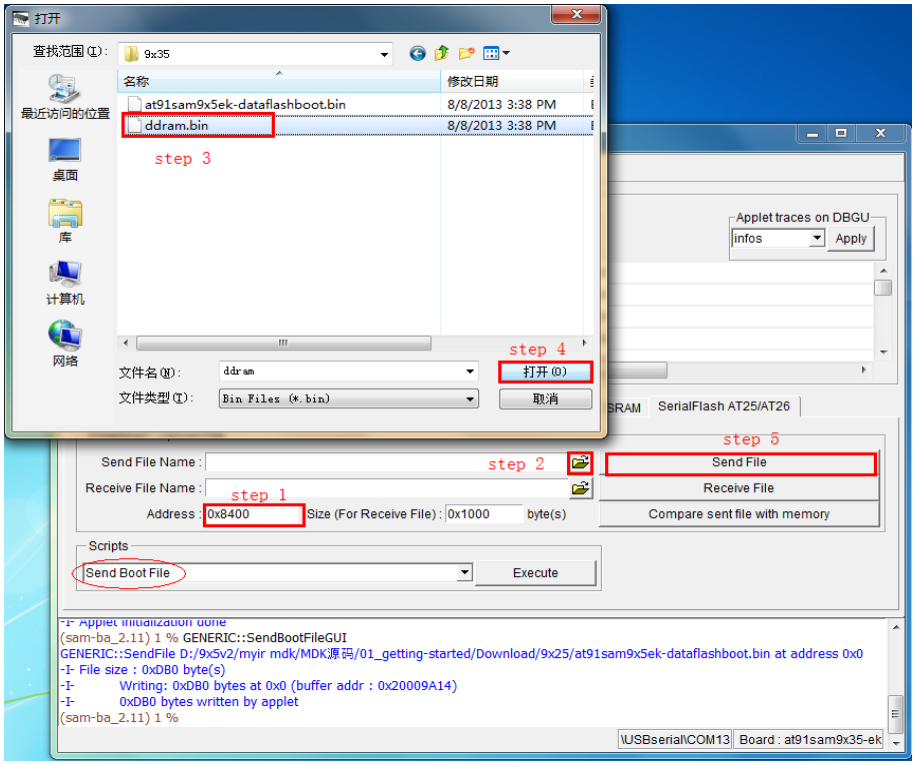


Figure 3-27

Note: Don't change the "Scripts" item, keep it the same as the previous step.

Lastly after sending "ddram.bin", press NRST to reset board (Open the terminal and configure the parameters beforehand, refer to Chapter 3.2.3), there will be terminal information and two lights flash alternately. Press character "1" the first time, only the red light on; press this character again, two lights flash alternately.

The above description is the whole processes of manual download.

3.2.5 Automatic Download

The example of "getting-started" describes the steps of automatic download. (Location: "04-MDK_Source\01_getting-started\Download\9x35"). Directory files are shown in Figure 3-28:

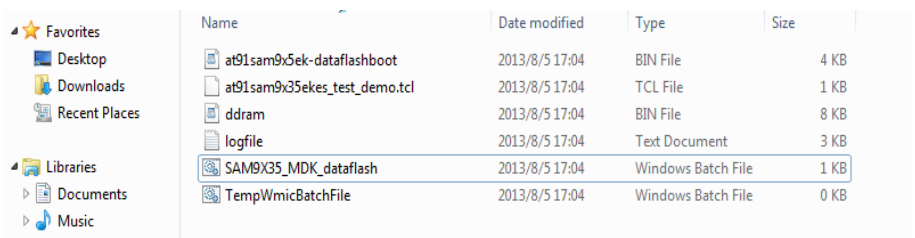


Figure 3-28

Right click to edit “SAM9X35_MDK_dataflash.bat”, then modify the COM port, refer to Figure 3-29:

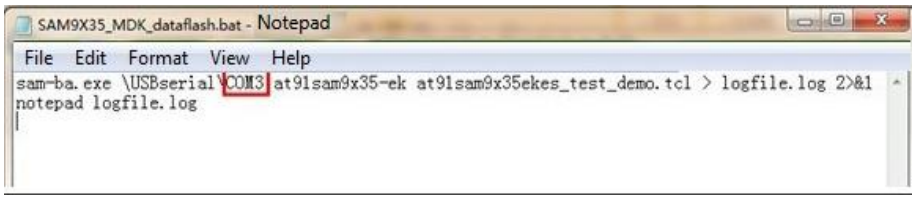

SAM9X35_MDK_dataflash.bat - Notepad
File Edit Format View Help
sam-ba.exe \USBserial\COM3 at91sam9x35-ek at91sam9x35ekes_test_demo.tcl > logfile.log 2>&1
notepad logfile.log

Figure 3-29

The local port is COM13, so change COM3 to COM13 (Not knowing the actual port of your computer, please refer to Chapter 3.2.3) and save it (Note: Don’t change the others).

Then, turn SW1_1, SW1_2 off (Note: disconnect the base board JP8 jump line, otherwise the computer will not recognize the development board), press NRST to reset board. After reset it, turn SW1_2 on and double click “SAM9X35_MDK_dataflash.bat” to start the automatic download process. It will pop up a “logfile which records downloading steps and information. (Related information will be recorded likewise if download fails).

After finishing downloading, press “K1” to reset board, the program will start to run. At this point, the process of automatic download has completed. The automatic and manual download are doing the same thing, It is recommended to use the automatic download to save time.

3.3 MDK Routine Introduction

The MDK sample programs basically cover the test and usage of all kinds of on board devices.

Important Note: Please refer to Chapter 3.2 for download steps and terminal configuration. The following examples will no longer describe how to download and configure terminal. All sample programs test requires relevant preparations.

3.3.1 Getting-started

- **Purpose**

  This example demonstrates chip startup sequence and core peripherals.

- **Functional Description**

This program makes two LEDs blink at a fixed rate, then type “1” or “2” to control LEDs.

➢ **Procedures**

Download the program into board, press NRSRT to observe relevant terminal information. Press “1” in terminal to start or stop LED1 blinking, there will be “1 2” or all “2” in terminal; Press “2” in terminal to start or stop LED1 blinking, there will be “1 2” or all “1” in terminal.

➢ **Phenomenon Indicates**

Terminal information:

```
-- Getting Started Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Jul  9 2012 11:13:49 --
Configure PIT
                Configure TC.
Configure LED PIOs.
No push buttons, uses DBG key 1 & 2 instead.
Press 1 to Start/Stop the blue LED D1 blinking.
Press 2 to Start/Stop the green LED D2 blinking.
1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2
2 2 1 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1 2 2 1 2 2 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2
1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2
1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1
2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 2 2
```

3.3.2 Adc_adc10

➢ **Purpose**

This example demonstrates ADC peripheral.

➢ **Functional Description**

This example aims at demonstrating how to use ADC with/without DMA. When working with DMA, it works as a big buffer for ADC peripherals, the data will be stored immediately without interfering with CPU. The steps are:

- Initialize ADC with expected parameters
- Configure and enable interrupt for ADC
- Enable DMA reception

- Check the last converted channel in ADC interrupt handler if DMA is not used

➢ **Procedures**

Download the program into board, press NRSRT to observe relevant terminal information. Press “d” in terminal enable/disable DMA; character “s” changes channel mode; Digital 0-3 represent 4 trigger modes, respectively; three data show three AD sampling data value, respectively.

➢ **Phenomenon Indicates**

Terminal information:

```
ADC12 Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    1 2013 21:55:10 --



================================
- d: DMA Enable/Disable
- s: Channel sequence switch
- 0, 1,    2,          3: TRIGGER mode:
   SW EXT Periodic Continuous
   Refresh slow --> fast ....
================================
= DMA: Enabled; Trigger mode: 0
= Sequence: 09 00 02
================================
Vols(mV): #09:3290 #00:3287 #02:2925
```

3.3.3 Adc_touch-screen

➢ **Purpose**

This example demonstrates touchscreen events..

➢ **Functional Description**

This example firstly initializes LCD and touchscreen controller. Then let user do calibration. After calibration is done, the pen positions and pressure will be displayed in terminal when touching LCD.

➢ **Procedures**

The package can only be used with MYD-SAM9G15/G35/X35.

Download the program into board, press NRSRT to observe relevant terminal

information. After touching the dots on the LCD to calibrate touchscreen, the calibration results will be outputted in terminal and LCD. When touching LCD, the pen coordinate will be outputted in terminal. If the calibration doesn't succeed retry until success.

- **Phenomenon Indicates**

  Terminal information:

```
-- ADC_Touchscreen Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  1 2013 22:55:00 --
-I- I cache is already enabled.
-I- P0: (162,793)
-I- P1: (885,777)
-I- P2: (787,292)
-I- P3: (520,358)
-I- Slope: 1320, -2160
-I- TP: 285, 389 -> 143, 218
-W- X 143, Y 218; Diff -97, 82
-E- Error too big ! Retry...
-I- P0: (162,806)
-I- P1: (883,787)
-I- P2: (884,209)
-I- P3: (160,248)
-I- Slope: 1926, -2667
-I- TP: 521, 446 -> 238, 165
-I- Calibration successful !

Pressed(352, 89, 1942)
Move    (353, 89, 1962)
Release(353, 89)

Pressed(156,170, 1883)
Move    (303,132, 1929)
Release(303,132)
```

3.3.4 CAN

- **Purpose**

  The CAN example demonstrates CAN peripheral.

- **Functional Description**

  This example aims at testing the following CAN operations:

  - Simple CAN test: CAN1 Mailbox 5 of one board sends data to CAN1 Mailbox 1

on another board.

- Messages to One and the same Mailbox test. CAN1 Mailbox 5 and 6 of one board send data, (Mailbox 6 goes first), CAN1 Mailbox 2 on another board receives them, but the last one is discarded.
- Messages to One and the same Mailbox test.CAN1 Mailbox 6 and 5 of one board send data in sequence, with ID 0x40 and 0x41 that both data can received by CAN0 Mailbox 3. The last data will overwrite the previous one.

➢ **Procedures**

This example can only be used with MYD-SAM9X25 / MYD-SAM9X35 development board.

After download, connect CAN_H1 and CAN_L1 in two development board, respectively. Press NRST, there will be terminal information. Press '1' to send message in serial port terminal 1, press '2' to receive message in serial port terminal 2.

➢ **Phenomenon Indicates**

Serial port terminal 1

Press '1' to send message

```
-- CAN Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug 21 2013 16:07:30 --
- Test start, DBGU not available now
-I- 1:    210000
- CAN1 Sync OK
==================
Press '1' Send Maxbox
Press '2' Recieve Maxbox and Check Maxbox
-I- 1:40e00060
-I- 1:40a00040
-I- 1:40a00040
-I- 1:    900060

- Press any key to test again
```

Serial port terminal 2

Press '2' to receives message

```
-- CAN Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug 21 2013 16:08:58 --
- Test start, DBGU not available now
-I- 1:    210000
```

```
- CAN1 Sync OK
==================
Press '1' Send Maxbox
Press '2' Recieve Maxbox and Check Maxbox
-I- 1:20e00002
-I- 1:20a00006
- CAN1.1: Simple test data received
-- test PASSED
- CAN1.2: Messages to 1 Mailbox received
-- test PASSED
-I- 1:    e0000e
- CAN1.3: Messages to 1 Mailbox(OVR) received
-- test PASSED
-I- 1:    10000e

- Press any key to test again
```

3.3.5 DMA

- **Purpose**

  This example demonstrates Atmel's AT91SAM9X5 microcontrollers.

- **Functional Description**

  This example demonstrates DMA data transfer. Switch multiple DMA buffers transfer by corresponding buttons.

- **Procedures**

  Download the program into board, press NRSRT to observe terminal information. 0-9, A, B are transmission choices of DMA buffer. "S" starts transporting and displays menu.

- **Phenomenon Indicates**

  Terminal information:

```
-- DMA Example 2.0 --
            -- SAM9XX5-EK
            -- Compiled: Aug    1 2013 23:25:34 --
Menu :
------
- 1-9, A, B: Programming DMAC for Multiple Buffer Transfers
      1: Single Buffer or Last buffer of a multiple buffer transfer
      2: Multi Buffer transfer with contiguous DADDR
      3: Multi Buffer transfer with contiguous SADDR
      4: Multi Buffer transfer with LLI support
```

```
        5: Multi Buffer transfer with DADDR reloaded
        6: Multi Buffer transfer with SADDR reloaded
        7: Multi Buffer transfer with BTSIZE reloaded and contiguous DADDR
        8: Multi Buffer transfer with BTSIZE reloaded and contiguous SADDR
        9: Automatic mode channel is stalling BTsize is reloaded
        A: Automatic mode BTSIZE, SADDR and DADDR reloaded
        B: Automatic mode BTSIZE, SADDR reloaded and DADDR contiguous
- s: Start DMA transfer
- h: Display this menu

Programming DMAC for Multiple Buffer Transfers in row 1
Programming DMAC for Multiple Buffer Transfers in row 2
Programming DMAC for Multiple Buffer Transfers in row 10
-I- Start DMA transfer
-I- The Source Buffer content before transfer
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00 02 04 06 08 0a 0c 0e 10 12 14 16 18 1a 1c 1e
00 03 06 09 0c 0f 12 15 18 1b 1e 21 24 27 2a 2d
00 04 08 0c 10 14 18 1c 20 24 28 2c 30 34 38 3c

-I- The Destination Buffer content before transfer
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a

-I- The Source Buffer content after transfer
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
00 02 04 06 08 0a 0c 0e 10 12 14 16 18 1a 1c 1e
00 03 06 09 0c 0f 12 15 18 1b 1e 21 24 27 2a 2d
00 04 08 0c 10 14 18 1c 20 24 28 2c 30 34 38 3c

-I- The Destination Buffer content after transfer
00 01 02 03 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a

Done
```

3.3.6 LCD

- **Purpose**

The 4.3-inch screen case is used to demonstrate how to configure LCD Controller (LCDC).

➢ **Functional Description**

This example configures LCDC for LCD to display and then draw test patterns on LCD.

➢ **Procedures**

This package can be used in MYD-SAM9G15/G35/X35-V2.

Download the program into board, press NRSRT to observe terminal information. Then test pattern is displayed on the LCD.

➢ **Phenomenon Indicates**

Terminal information:

```
-- LCD Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  1 2013 23:40:10 --
-I- I cache is already enabled.
- Test Pattern: 480 x 272 [80 x 68]
- Test Cursor: 32 x 32
- LCD ON
Show: 82,37 32x48 0
Show: 164,76 64x192 0
Show: 246,45 64x-192 0
Show: 328,6 -64x-192 0
Show: 410,31 -64x192 0
Show: 339,70 32x48 0
Show: 257,51 64x192 0
Show: 175,12 192x64 90
Show: 93,135 -192x64 90
Show: 11,80 64x192 180
Show: 70,42 192x64 270
Show: 152,165 192x-64 270
Show: 234,80 64x192 0
```

3.3.7 Periph _protect

➢ **Purpose**

This program demonstrates PIO controller behavior.

➢ **Functional Description**

This application shows protective mechanism of PIO controller. When the write-protection is enabled, any write attempt to write-protected registers is aborted. So register won't be modified. Besides, the write protect registers save register offset address.

- **Procedures**

Download the program into board, press NRSRT to observe relevant terminal information. Typing "I" in terminal will enable write-protect mode, while typing "U" will enable unprotected mode.

- **Phenomenon Indicates**

Terminal information:

```
-- Peripheral Protect Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 09:06:13 --


Enter 'I' to enable Write Protect and enter 'u' to disable Write Protect.

Select the register to be written by a value(0x12345678).
  0 : PIO Enable Register          (0x0000)
  1 : PIO Disable Register         (0x0004)
  2 : PIO Output Enable Register           (0x0010)
  3 : PIO Output Disable Register          (0x0014)
  4 : PIO Input Filter Enable Register   (0x0020)
  5 : PIO Input Filter Disable Register (0x0024)
  6 : PIO Multi-driver Enable Register   (0x0050)
  7 : PIO Multi-driver Disable Register (0x0054)
  8 : PIO Pull Up Disable Register         (0x0060)
  9 : PIO Pull Up Enable Register          (0x0064)
  a : PIO Peripheral ABCD Select Register 1        (0x0070)
  b : PIO Peripheral ABCD Select Register 2        (0x0074)
  c : PIO Output Write Enable Register   (0x00A0)
  d : PIO Output Write Disable Register (0x00A4)
  e : PIO Pad Pull Down Disable Register           (0x0090)
  f : PIO Pad Pull Down Enable Register (0x0094)


No write protect violation is detected.
The Write Protect is enabled.
Write protect violation is detected!
The offset of the write-protected register is 0x0000.
```

```
Write protect violation is detected!
The offset of the write-protected register is 0x0004.
Write protect violation is detected!
The offset of the write-protected register is 0x0070.
Write protect violation is detected!
The offset of the write-protected register is 0x0094.
The Write Protect is disabled.
No write protect violation is detected.
No write protect violation is detected.
```

3.3.8 Pmc_clock_switching

- **Purpose**

  This example demonstrates switch system clock (PLLA, UPLL, SLCK, MAINCK).

- **Functional Description**

  Upon startup, the program configures PIOs for DBUG, PCK. DBUG baud rate is configured as 1200 bps. This example prints the current configuration and waits input to switch system clock.

- **Procedures**

  This program is different from others. Firstly, DBGU baud is configured as 1200 bps, while others don't change. Secondly, manual download has a little change, please accord with the following steps:

  Turn SW1_1, SW1_2 off, press NRST and open samba 2.11 (The same as manual download in Chapter 3.24). Then turn SW1_2 on, enable SerialFlash and download "sram.bin" file. Specific operations are shown in Figure 3-30, Figure 3-31:

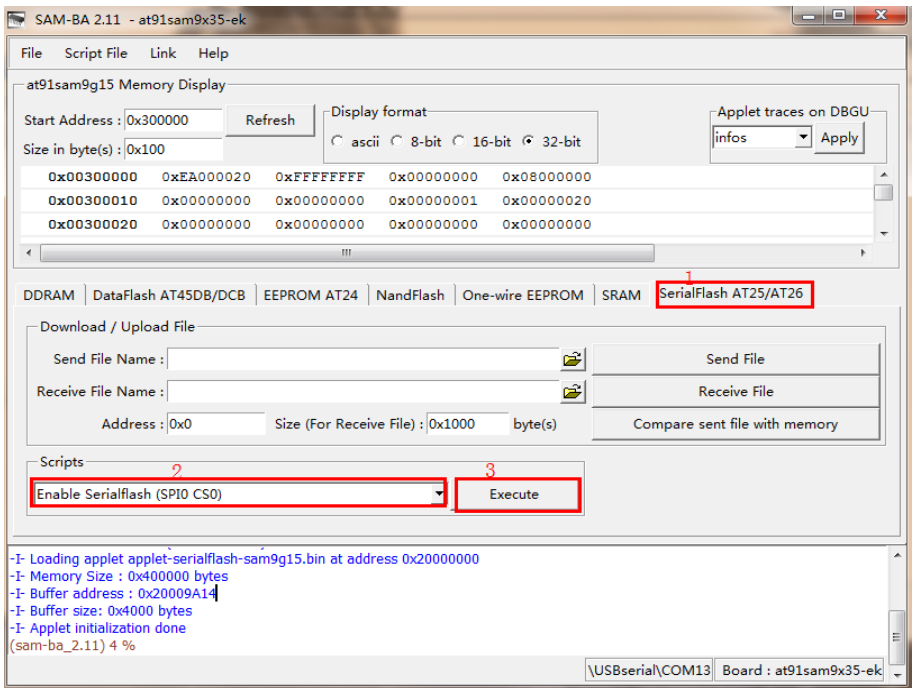


Figure 3-30

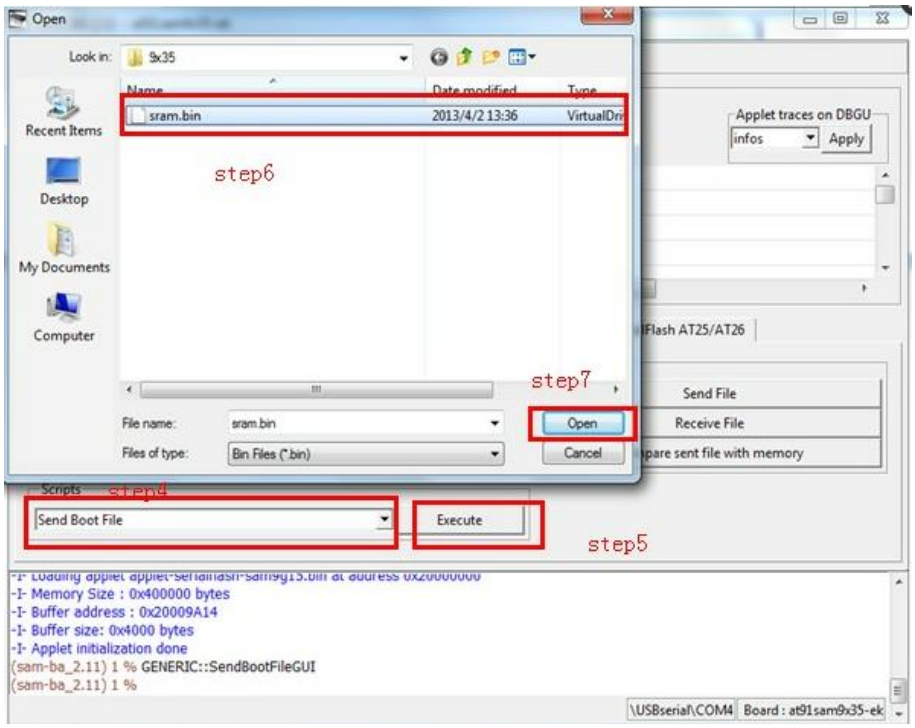


Figure 3-31

Download the program into board, press NRSRT to observe relevant terminal information, and switch system clock according to prompts.

- **Phenomenon Indicates**

Terminal information:

```
** Switch to 1200 bps for DBG **
```

```
-- PMC Clock Switching example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    2 2013 09:12:21 --

 --- Current PMC clock from lowlevel pmc configuration ---
The slow clock source is internal 32 kHz RC oscillator
PLLA clock is 800 MHz
PLLA clock is the source of Master clock
MCK Master Clock is prescaler output clock divided by 3

-I- Select main clock as the master clock
-I- Please measure the clock on PCK to make sure it is 12000000 Hz...
-I- Press ` to switch next clock configuration...

-I- Select PLLA clock as the master clock
-I- Please measure the clock on PCK to make sure it is 12500000 Hz...
-I- Press ` to switch next clock configuration...

-I- Select UTMI PLL clock as the master clock
-I- Please measure the clock on PCK to make sure it is 7500000 Hz...
-I- Press ` to switch next clock configuration...

-I- Switch the XTAL 32K crystal oscillator to be the source of the slow clock
-I- Please measure the clock on PCK to make sure it is 32768 Hz...
-I- Debuging in EWARM IAR C_SPY, the JLINK will disconnect on some PC!
-I- Press ` to switch next clock configuration...


Done
```

3.3.9 PWM

- **Purpose**

  This example demonstrates PWM channel configuration.

  A simple demonstration of two PWM channels configuration is used to produce variable cycle and frequency of signal. When connecting PB18, PWM0 (PC10) or PB18 and PWM1 (PC11) will make two LEDs flashing.

- **Functional Description**

  Two PWM channels (channel #0, #1) are configured to generate two PWM signals.

- **Procedures**

Download the program into board, press NRSRT to observe relevant terminal information.

(1) Connecting pin 7 in J5 to pin 8 in J2, blue LED start glowing repeatedly at f1.

(2) Connecting pin 7 in J5 to pin 12 in J2, blue LED start glowing repeatedly at f2.

- **Phenomenon Indicates**

Two different connection ways lead the LEDs flashing at different frequencies.

3.3.10 Ssc_dma_audio

- **Purpose**

This example demonstrates outputting an audio stream by WM8731CODEC.

- **Functional Description**

This example plays a pre-loaded WAV file in flash. The audio stream is outputted by WM873 SSC port. Audio format:

Format: WAV

Sample Rate: 48 kHz

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information:

```
-- ssc_dma_audio --
Menu :
------
x: Receive WAV file with XModem Protocol
   X: Receive WAV file through DBGU
```

Input “x” in terminal

```
Transfer wav file with 1K XModem, Ctr+ D to cancel
```

Operation can be done in Figure 3-32 and Figure 3-33.

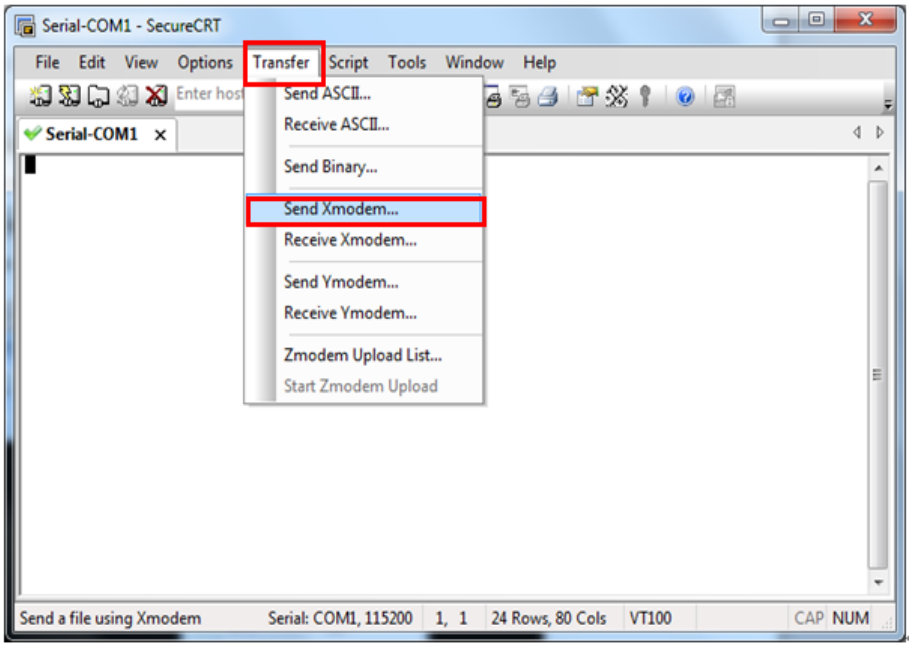


Figure 3-32

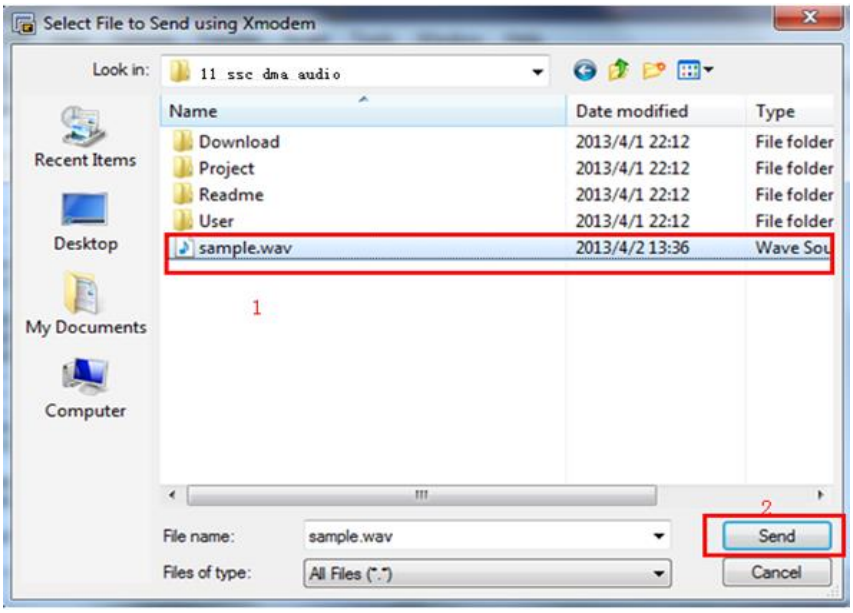


Figure 3-33

Starting “x modem” transmission after clicking “OPEN”, press “Ctrl+C” to cancel transfer and wait for end prompt:

```
100%       274 KB      5 KB/s 00:00:49          0 Errors
-- ssc_dma_audio --
Menu :
------
  W: Play the WAV file loaded
    I: Display the information of the WAV file
    x: Receive WAV file with XModem Protocol
    X: Receive WAV file through DBGU
```

Input 'W' to choose WAV file and "I" to output audio:

```
-- ssc_dma_audio --
Menu :
------
W: Play the WAV file loaded
I: Display the information of the WAV file
```

Pressing "W" mounts WAV audio and the terminal display:

```
-- ssc_dma_audio --
Menu:
------
    I: Display the information of the WAV file
   S: Stop playback
```

Pressing "I "outputs Audio:

```
-- WAV file @ 22000000
Wave file header information ------------------------------
  - Chunk ID            = 0x46464952
  - Chunk Size          = 281028
  - Format              = 0x45564157
  - SubChunk ID         = 0x20746D66
  - SubchunNRST Size    = 16
  - Audio Format        = 0x0001
  - Num. Channels       = 2
  - Sample Rate         = 48000
  - Byte Rate           = 192000
  - Block Align         = 4
  - Bits Per Sampl= 16
  - Subchunk2 ID        = 0x61746164
  - Subchunk2 Size      = 280992

-- Press any key to return to menu
```

➢ **Phenomenon Indicates**

Terminal outputs information in detail and headphone outputs audio.

3.3.11 Twi_eeprom

➢ **Purpose**

This example program demonstrates TWI peripheral accessing an external serial EEPROM chip.

➢ **Functional Description**

This example is used to test EEPROM model.

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information:

- **Phenomenon Indicates：**

Terminal information:

```
-- TWI EEPROM Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 10:35:30 --
-I- Filling page #0 with zeroes ...
-I- Filling page #1 with zeroes ...
-I- Read/write on page #0 (polling mode)
-I- 0 comparison error(s) found
-I- Read/write on page #1 (IRQ mode)
-I- Callback fired !
-I- Callback fired !
-I- 0 comparison error(s) found
```

3.3.12 Usart_serial

- **Purpose**

This example demonstrates USART simulating DBUG.

- **Functional Description**

On startup, the debug information is printed by DBGU port. USART0 will send back any character it receives from the HyperTerminal as well as text file.

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information:

```
-- USART Serial Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 11:15:21 --
-- Start to echo serial inputs --
```

Unplug the serial cable from DBUG (J18) and insert UART0 (J16). The terminal displays:

```
Start waiting data by using DMA:
```

(1) At this point, pressing the key board will echo corresponding characters:

```
fffndfgsgw2hdf4s15xc0dsgdfddddddddd
```

(2) Send a txt document. Build a text document and send it. Refer to Figure 3-34 and Figure 3-35:

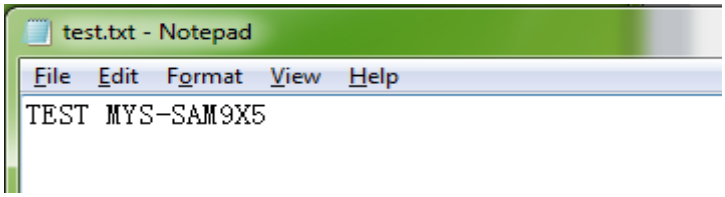


Figure 3-34

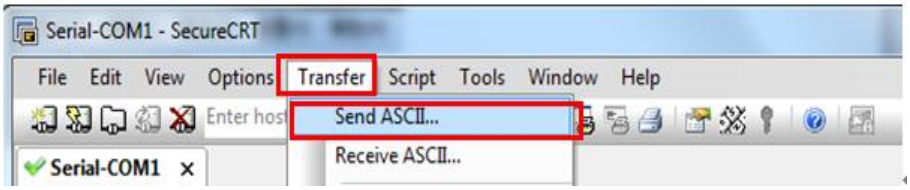


Figure 3-35

Terminal information:

```
Start waiting data by using DMA:
fffndfgsgw2hdf4s15xc0dsgdfddddddddd
TEST MYS-SAM9X5
```

➢ **Phenomenon Indicates**:

The phenomenon has explained as above.

3.3.13 Emac0

➢ **Purpose**

This example demonstrates Ethernet MAC (EMAC) and Ethernet transceiver.

➢ **Functional Description**

Upon startup, configure board by default IP (192.168.2.115) and MAC address, test IP by ping command.

➢ **Procedures**

This example can be used with MYD-SAM9G25/G35/X25/X35-V2 and requires Network port J16.

(1) Connect board to network or to PC by crosswire. Then set the host IP 192.168.2.XX (Note: XX can't be 115).

(2) Download program into board, press NRSRT to observe relevant terminal information:

(3) Open a terminal application and type the following command line:

```
ping 192.168.2.115.
```

➢ **Phenomenon Indicates**

Terminal information:

```
-- -- EMAC Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 16:31:49 --
-- MAC 0:45:56:78:9a:bc
-- IP  192.168.2.115
-I- ** Valid PHY Found: 0
-I- AutoNegotiate complete
P: Link detected
```

Input the command in terminal: ping 192.168.2.115. Refer to Figure 3-36:

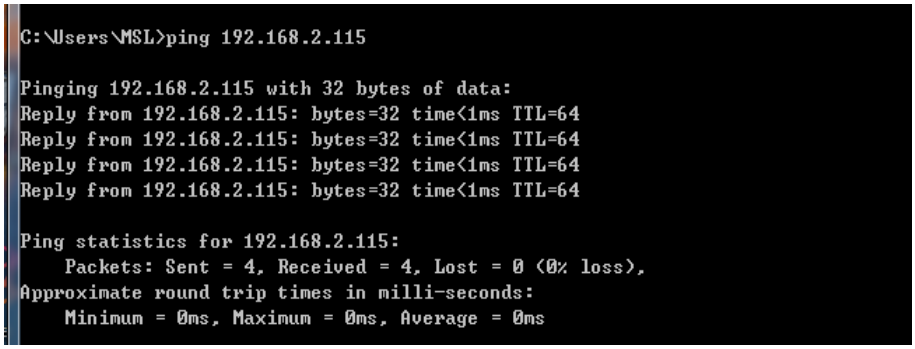

C:\Users\MSL>ping 192.168.2.115
Pinging 192.168.2.115 with 32 bytes of data:
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.2.115:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

Figure 3-36

3.3.14 Emac1

➢ **Purpose**

This example demonstrates Ethernet MAC (EMAC) and Ethernet transceiver.

➢ **Functional Description**

Upon startup, configure board by a default IP (192.168.2.115) and MAC address, test IP by ping command.

➢ **Procedures**

The program can be used with MYD-SAM9X25-V2 and requires Network port J15 (connect JP3 to enable J15).

(1) Connect board to network or to PC by crosswire. Then set the host IP 192.168.2.XX (Note: XX can't be 115).

(2) Download program into board, press NRSRT to observe relevant terminal information.

(3) Open terminal and type the following command line:

```
ping 192.168.2.115.
```

➢ **Phenomenon Indicates**

Terminal information:

```
-- EMAC Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 16:50:04 --
-- MAC 0:45:56:78:9a:bc
-- IP  192.168.2.115
-I- ** Valid PHY Found: 0
-I- AutoNegotiate complete
P: Link detected
```

Input the command in terminal: ping 192.168.2.115. Refer to Figure 3-17:

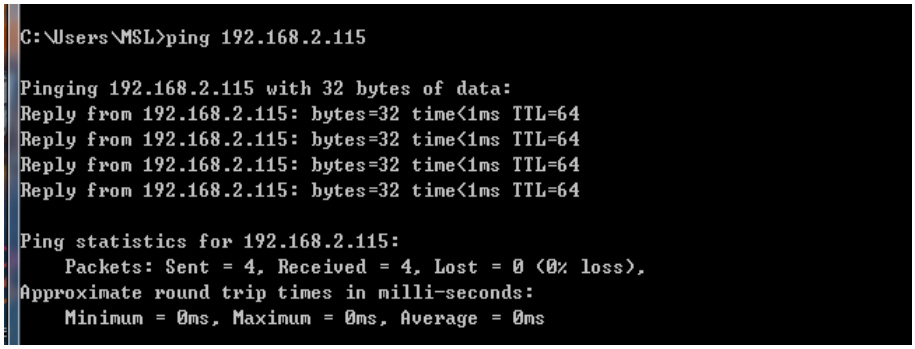

C:\Users\MSL>ping 192.168.2.115
Pinging 192.168.2.115 with 32 bytes of data:
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Reply from 192.168.2.115: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.2.115:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms

Figure 3-37

3.3.15 Hsmci_multimedia_card

➢ **Purpose**

This example demonstrates HSMCI port on SAM microcontrollers.

➢ **Functional Description**

Open HyperTerminal before running this program. On starting this program, the HyperTerminal will print test information which includes initialization and performance.

➢ **Procedures**

Download program into board, press NRSRT to observe relevant terminal information, and then insert a SD card.

➢ **Phenomenon Indicates**

Terminal information:

Without a SD card:

```
-- Basic MultiMedia Card Project 2.0 --
-- SAM9XX5-EK
```

```
-- Compiled: Aug    2 2013 17:10:56 --
-I- Cannot check if SD card is write-protected
```

With insert a SD card:

```
=========================================
=========================================
-I- SdMmcIdentify.Cmd5: 3
-E- Acmd41.cmd55:3
-E- SdMmcIdentify.Cmd1: 3
-E- SD_Init.Identify: 5
-I- SdMmcIdentify.Cmd5: 3
-I- SD MEM
-I- Card Type 2, CSD_STRUCTURE 0
-W- SD 4-bit mode
-I- HS Not Supported in SD Rev 0x0
-I- Set SD/MMC clock to 22222K
-I- SD/MMC card initialization successful
```

Press Enter:

```
=========================================
-!- MCK is 133MHz
-!- Buffer@2000b76c,size 0x400000
# i,I     : Re-initialize card
# t       : Disk R/W/Verify test
# T       : Disk performance test
# p       : Change number of blocks in one access for test
# m       : Change MCI port used
```

Input the corresponding command for different operations. Input ‘t”:

```
=========================================
-!- Test code: 1.clr, 2.wr, 3.rd
-I- Testing block [ 37176 -    41271] …
```

3.3.16 Hsmci_sdcard

- **Purpose**

  This example demonstrates HSMCI port.

- **Functional Description**

  This example detects, initializes SD/MMC memory card, and performs R/W test on it.

- **Procedures**

  Download program into board, press NRSRT to observe relevant terminal information.

When prompted “Please insert a card”, the program starts to initialize and test SD card.

- **Phenomenon Indicates**

Terminal information:

```
-- Basic HSMCI SD/MMC Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    2 2013 17:16:06 --
-I- Cannot check if SD card is write-protected
-- Please insert a card

=========================================
-I- SdMmcIdentify.Cmd5: 3
-I- SD MEM
-I- Card Type 2, CSD_STRUCTURE 0
-W- SD 4-bit mode
-I- HS Not Supported in SD Rev 0x0
-I- Set SD/MMC clock to 22222K
-I- SD/MMC card initialization successful
-I- Card size: 124 MB, 249344 * 512B
…
=========================================
-!- MCI 0, code: 1.clr, 2.wr, 3.rd
-I- Testing block [ 97566 -    97569] ...
```

3.3.17 Smc_nandflash

- **Purpose**

  This example demonstrates s read/write data from/to NandFlash SMC.

- **Functional Description**

  This sample program tests the NandFlash module of the development board, which needs to configure the NandFlash port with SMC. Inputting the corresponding command according to prompt, result message will display on terminal.

- **Procedures**

  Download program into board, turn SW1_1 on, press NRSRT to observe relevant terminal information.

- **Phenomenon Indicates**

  Terminal information:

```
-- SMC NandFlash Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    2 2013 17:24:12 --
-I- Nandflash ID is 0x9580DA2C
Menu :
```

```
------
- i: Dump Nand flash information
- d: Enable or disable DMA
- r: Performance test (Raw without ECC)
- s: Performance test (Software ECC)
- p: Performance test (PMECC)
- h: Display this menu
```

Input “I”:

```
-I- Size of the whole device in bytes : 0x10000000
-I- Size in bytes of one single block of a device : 0x20000
-I- Number of blocks in the entire device : 0x800
-I- Size of the data area of a page in bytes : 0x800
-I- Number of pages in one block : 0x40
```

Input “d”:

```
-I- Initialize DMA done.
-I- Disable DMA done.
-I- Initialize DMA done.
-I- Disable DMA done
```

Input “r”:

```
-I- Erase block 10
-I- Write block 10
-I- Raw block write speed 4228K/s
-I- Read block 10
-I- Raw block Read speed 6553K/s
Menu :
------
- i: Dump Nand flash information
- d: Enable or disable DMA
- r: Performance test (Raw without ECC)
- s: Performance test (Software ECC)
- p: Performance test (PMECC)
- h: Display this menu
```

Input “s”:

```
-I- Disable PMECC using Software ECC.
-I- Erase block 10
-I- Write block 10
-I- Raw block write speed 1506K/s
-I- Read block 10
-I- Raw block Read speed 1899K/s
Menu :
------
- i: Dump Nand flash information
```

```
- d: Enable or disable DMA
- r: Performance test (Raw without ECC)
- s: Performance test (Software ECC)
- p: Performance test (PMECC)
- h: Display this menu
```

Input “p”:

```
-I- Initialize PMECC.
-I- Erase block 10
-I- Write block 10
-I- Raw block write speed 3542K/s
-I- Read block 10
-I- Raw block Read speed 5242K/s
Menu :
------
- i: Dump Nand flash information
- d: Enable or disable DMA
- r: Performance test (Raw without ECC)
- s: Performance test (Software ECC)
- p: Performance test (PMECC)
- h: Display this menu
```

Inputting “h” will display menu.

3.3.18 Spi_serialflash

- **Purpose**

This example demonstrates how to set up SPI and how to read/write serial dataflash.

- Functional description

This example tests serial data flash by erasing/writing each pages, and read/write bandwidth.

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information. (Note: Turn SW1_2 on)

- **Phenomenon Indicates**

Terminal information:

```
-- SPI with Serialflash Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 17:35:23 --
DMA driver initialized with IRQ
```

```
SPI and AT25 drivers initialized
ID read: 1471f
AT25DF321A serial flash detected
Flash unprotected
Chip is being erased...
```

After a certain period of time:

```
Checking erase ...
Checking page #16383
Erase successful.
Programming a walking 1 on all pages ...
Programming page #16383
Walking 1 test successful.
```

3.3.19 Usb_audio_looprec

- **Purpose**

This example demonstrates UDP and DACC on AT91SAM microcontrollers, as well as USB framework.

- **Functional Description**

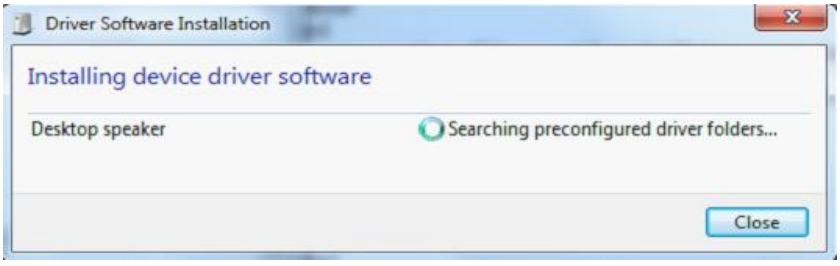
Input loop back sound to a simulated USB Desktop Speaker, connect board to the host by USB cable, and then play music, the audio stream is sent to board. At the same time, the audio stream received is also sent back to the host for recording.

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information, connect PC by USB cable, the host reports a new USB device attachment.

- **Phenomenon Indicates**

Start board, the host will report a new USB device attachment and install it automatically. “USB Audio Device” appears in hardware device list. Refer to Figure 3-38:



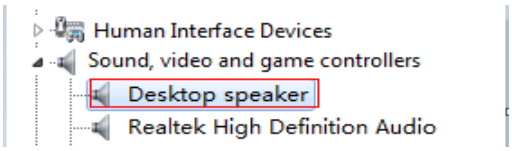


Figure 3-38

Terminal information:

```
-- USB Device Audio LoopREC Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  2 2013 17:48:31 --
USBD_Init
```

3.3.20 Usb_cdc_serial

- **Purpose**

This example demonstrates USB Device Port (UDP) and USART port on AT91SAM microcontrollers, as well as USB Framework.

- **Functional Description**

This demo simulates a USB to RS-232 Serial Port Converter

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information, the host will report a new USB device attachment (Note: it may not appear in some computers). There will be additional serial port after installing driver.

- **Phenomenon Indicates**

There will be device in Figure 3-39:

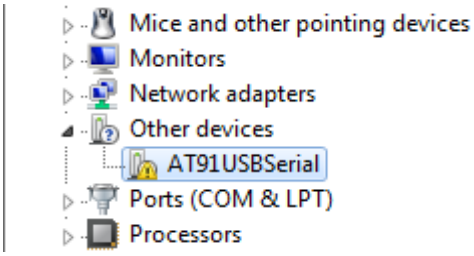


Figure 3-39

Updating driver (location: 04_MDK_Source\libraries\usb\device), there will be additional “AT91 USB to Serial Converter (COM4)” in the device. Refer to Figure 3-40:

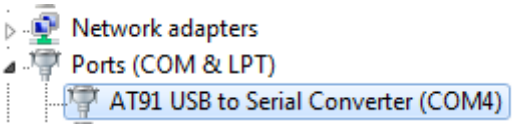


Figure 3-40

Terminal information:

```
-- USB Device CDC Serial Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 10:24:06 --
-I- CDCDSerial_Initialize
-I- CDCDSerialPort_Initialize
USBD_Init
-- ESC to Enable/Disable ECHO on cdc serial --
-- TAB to Enable/Disable DEBUG log output    --
-I- VBus configuration
-I- connn
```

3.3.21 Usb_core

- **Purpose**

  This example demonstrates UD port on AT91SAM microcontroller.

- **Functional Description**

  Connect PC by USB cable, and the host will prompt USB device attachment.

- **Procedures**

  Download program into board, press NRSRT to observe relevant terminal information, connect to PC by USB cable, the host reports a new USB device attachment. (Note: some computers may not report it). The device manger is shown in Figure 3-38:

- **Phenomenon Indicates**

  New hardware USB device can be found in device manger. Refer to Figure 3-41:

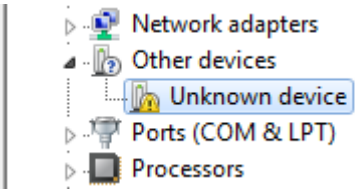


Figure 3-41

Terminal information:

```
-- USB Device Core Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 10:26:19 --
-I- USB initialization
USBD_Init
-I- Connecting device
-I- VBus configuration
```

```
    -I- conn
    Rsm Susp Rsm Std gDesc Dev Std sAddr SetAddr(1) Std gDesc Dev Std gDesc Cfg Std gDesc Cfg Std gDesc Str238 Std gDesc Dev Std sAddr SetAddr(2) Std gDesc Dev Std gDesc Cfg
```

3.3.22 Usb_hid_key board

➢ **Purpose**

This example demonstrates UDP and PIO port on AT91SAM microcontrollers, USB Framework is used for USB driver such as USB HID.

➢ **Functional Description**

This example simulates a simple key board. Connect the board to the host by USB cable, the host reports a new hardware attachment. Refer to Figure 3-42:

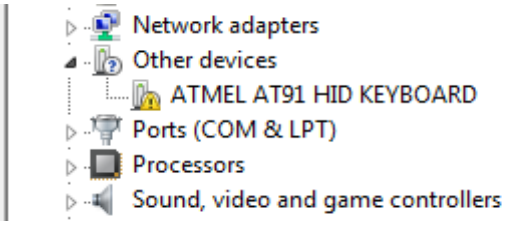


Figure 3-42

After installing driver, new USB Device appears in the hardware device list. As shown in Figure 3-43:

Figure 3-43

➢ **Procedures**

Download program into board, press NRSRT to observe relevant terminal information, PC reports new device and installs it automatically. After installing, a new USB device is added to human input device in the device manger.

➢ **Phenomenon Indicates**

A new USB device is added in device manger. Refer to Figure 3-39:

Terminal information:

```
-- USB Device HID Key board Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 10:37:52 --
-- : DBG key 1 2 used as buttons
-- : 1st press to push, 2nd press to release
-I- HIDDFunction_Initialize
USBD_Init
-I- VBus configuration
-I- con
```

Typing “1” makes terminal print character “a” continuously, while typing ”1” again makes terminal stop printing character “a”.

```
-I- Key 0 pressed
aaaa
```

New USB device can be found in device manger. Refer to Figure 3-44:

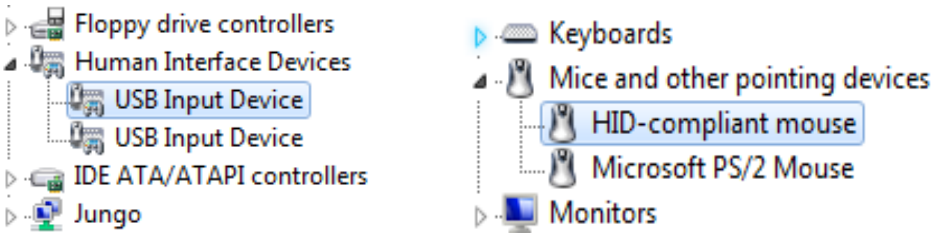


Figure 3-44

Terminal information:

```
-- USB Device HID Mouse Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    3 2013 10:49:28 --
-- Press W S A D to move cursor
-I- HIDDFunction_Initialize
USBD_Init
-I- VBus configuration
-I- conn
```

Pressing the buttons “WSAD” on the key board can move the host cursor.

3.3.24 Usb_hid_msd

- **Purpose**

This example demonstrates USB Device Port (UDP) port and other ports, as well as USB Framework.

- **Functional Description**

This example simulates a USB device integrating a key board and a removable disk. Connect the board to the host by USB cable, and the host will notice USB device attachment. After installing, there is a 10M removable disk.

- **Procedures**

Download program into board, press NRSRT to observe relevant terminal information. Connect the board to the host by USB cable, and the host will notice USB device attachment. After installing it, there is a 10M removable disk.

- **Phenomenon Indicates**

After starting board, a new USB device can be found in the device manger. Refer to Figure 3-45:

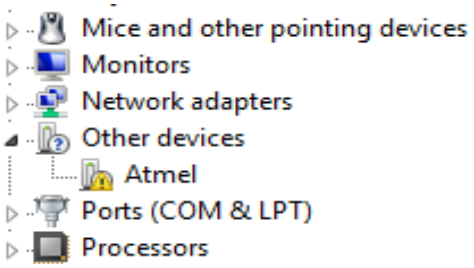


Figure 3-45

After installing driver, there will pop up a dialog box inquiring whether or not to format the disk. Refer to Figure 3-46:

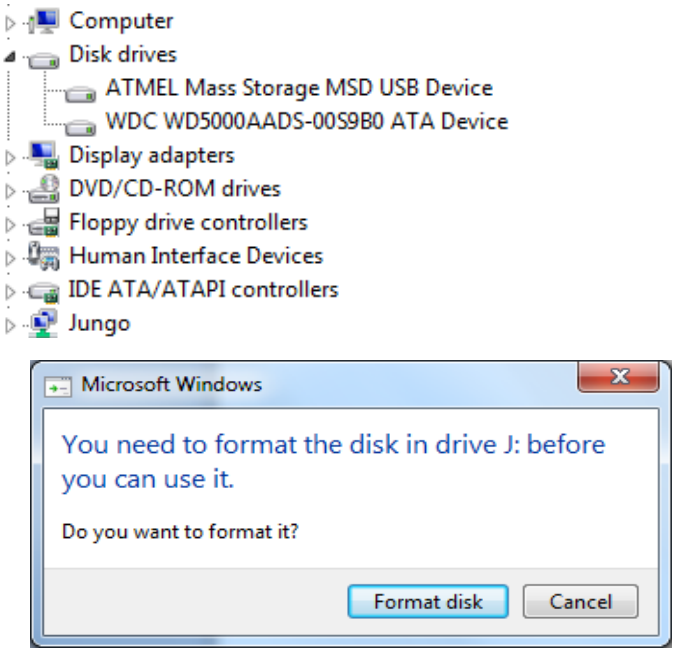


Figure 3-46

Choose to format the disk and it pops up dialog of formatting removable disk. Refer to Figure 3-47:

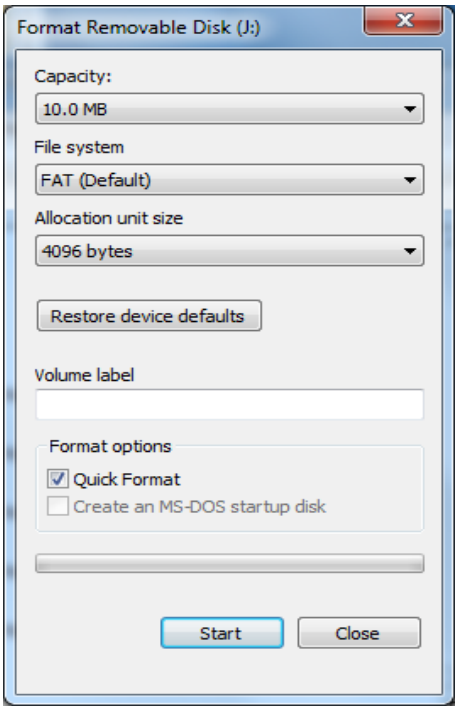


Figure 3-47

After formatting it, there is a 10M removable disk. Refer to Figure 3-48:

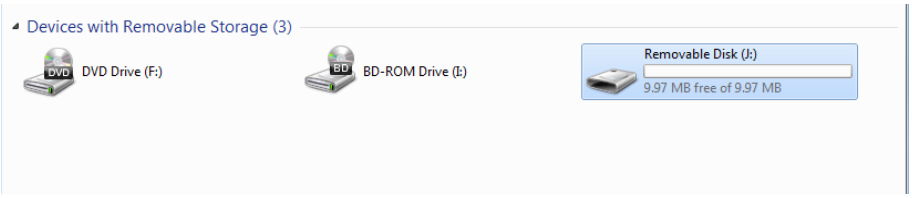


Figure 3-48

Terminal information:

```
-- USB HIDMSD Device Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 11:08:31 --
-- : DBG key 1 2 used as buttons
-- : 1st press to push, 2nd press to release
-I- LUN init
RamDisk @ 22000000, size 10485760
-I- RAM Disk init
-I- LUN init
-I- LUN: blkSize 1, size 20480
-I- HIDDFunction_Initialize
-I- MSDFun init
MSDReset USBD_Init
-I- VBus configuration
-I- conn
------------------------------
Inquiry Sending
```

```
Inquiry Sent Cplt
SendCSW ok
------------------------------
Inquiry Sending
Inquiry Sent Cplt
SendCSW ok
------------------------------
-W- MSDD_ProcessCommand: Unknown cmd 0x23
StaIN Cplt StallIn WaitHALNewReq Kbd T
SendCSW ok
------------------------------
ReqSense
ReqSense Cplt
SendCSW ok
------------------------------
.....（More information is not a comprehensive display）
```

3.3.25 Usb_hid_transfer

➢ **Purpose**

This example demonstrates the UDP and PIO port on AT91SAM microcontrollers, as well as USB Framework that is used for USB drivers such as HID.

➢ **Functional Description**

This example simulates a customized HID device that includes customized data stream of LEDs and buttons. Connect the board to the host by USB cable, and the host will report a new device attachment.

➢ **Procedures**

(1) Download program into board, press NRSRT to observe relevant terminal information.

(2) Connect board to PC by USB cable, the LED blinks. Open hidTest.exe (Location: 04_MDK_Source\25_usb_hid_transfer\hidTest.exe) to test new device information

(3) Find HID Device whose VID is 03EB and PID is 6201, choose item type and item to see its attributes.

(4) Input anything you want to the output edit box and send it, the information will display on terminal.

Phenomenon Indicates:

After starting board, the host will report a new device attachment and install driver automatically. Refer to Figure 3-49:

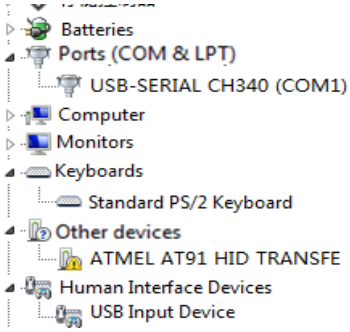


Figure 3-49

After installing driver, a new USB device and a HID-compliant device are added to human input device. Refer to Figure 3-50:

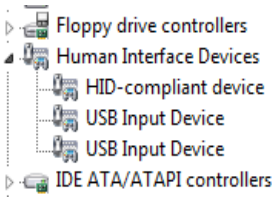


Figure 3-50

Open software (04_MDK_Source\25_usb_hid_transfer\hidTest.exe) and click “Read” to read HID ID. Click LED1 and LED2 which control the blue light and red light, respectively. Refer to Figure 3-51:

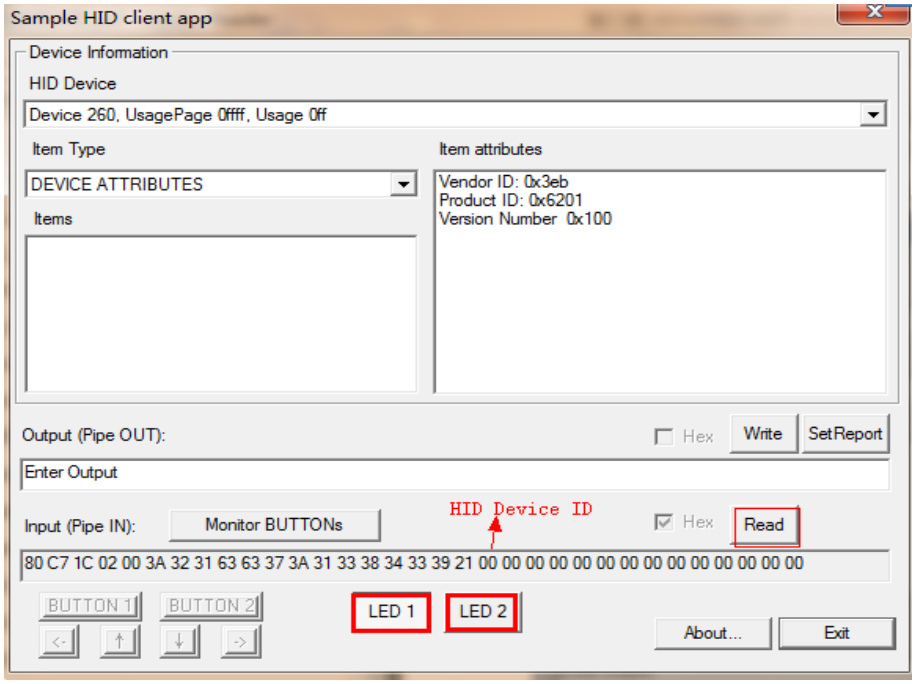


Figure 3-51

Terminal information:

```
-- USB Device HID Transfer Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 11:21:34 --
-- : DBG key 1 2 used as buttons
-- : 1st press to push, 2nd press to release
-I- HIDDFunction_Initialize
USBD_Init
-I- VBus configuration
-I- conn
Data In(32):
 81 00 00 1e 00 00 00 50
 02 ff 73 66 de fb 73 00
 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00
Data In(32):
 83 00 00 1e 00 00 00 50
 02 ff 73 66 de fb 73 00
 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00
Data In(32):
 82 00 00 1e 00 00 00 50
 02 ff 73 66 de fb 73 00
 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00
Data In(32):
 80 00 00 1e 00 00 00 50
 02 ff 73 66 de fb 73 00
 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00
```

3.3.26 Usb_iad_cdc_cdc

- **Purpose**

This example demonstrates UDP port and other ports, USB Framework that is used for USB drivers such as USB CDC, as well as combination between two USB and single composite device (such as Dual CDC port).

- **Functional Description**

This example simulates USB to RS-232 Serial Port Converter. Connect the board to

the host by USB cable, and the host will notice USB device attachment.

➢ **Procedures**

Download program into board, press NRSRT and observe relevant terminal information. When connecting to PC by USB cable, LED blinks and the host reports a new USB device attachment. Send data to port and observe it in another HyperTerminal connected to USART or USB.

➢ **Phenomenon Indicates**

After starting board, the host will report a new device attachment and install driver automatically.

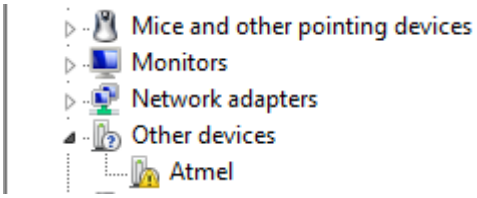


Figure 3-52

If the driver has not installed successfully (in Figure 3-52), install it manually (Location: 04_MDK_Source\libraries). The installation method can refer to Chapter 3.2.3. If the installation is completed, there will be two new COM devices. Refer to Figure 3-53:

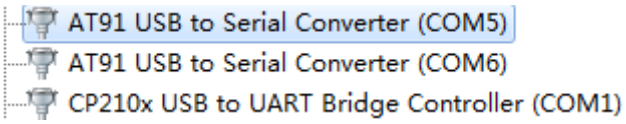


Figure 3-53

Terminal information:

```
-- USB Dual CDC Device Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 15:46:56 --
-I- DUALCDCDDriver_Initialize
-I- CDCDSerialPort_Initialize
-I- CDCDSerialPort_Initialize
USBD_Init
-I- VBus configuration
-I- conn
```

3.3.27 Usb_iad_cdc_hid

➢ **Purpose**

This example demonstrates USB Device Port (UDP) port and other ports as well as USB Framework that is used for USB driver such as USB CDC, and combination between two USB and a single composite (such as CDC+HID).

➢ **Functional Description**

This example simulates a USB composite device that has USB to Serial RS232 Converter and USB HID Key board functions. When connect the board to the host by USB cable, the host will notice USB device attachment. After installing driver, device manger adds COM and key board devices.

➢ **Procedures**

Download program into board, press NRSRT to observe relevant terminal information. When connect to PC by USB cable, the LED blinks and the host reports a new USB device attachment. After installing driver, “AT91 USB to Serial Converter” and “HID key board Device” are added to device manger. Typing “1” makes terminal continuously print “a”, while typing “1” again makes terminal stop printing “a”.

➢ **Phenomenon Indicates**

After starting board, the host will report a new device attachment and install driver automatically. After installing driver, a new device can be found in device manger. Refer to Figure 3-54:

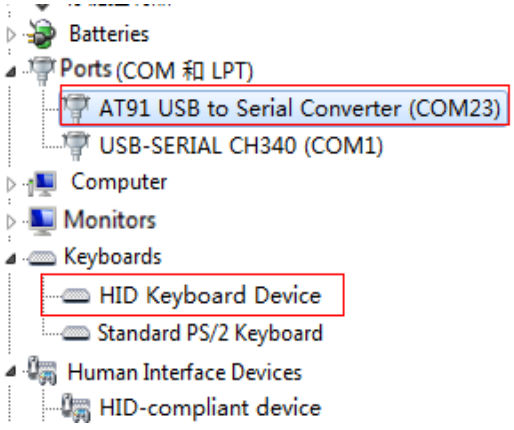


Figure 3-54

Note: If the driver has not installed successfully (in Figure 3-49), install it manually (Location: 04_MDK_Source\libraries). The installation method can refer to Chapter 3.23.

Terminal information:

```
-- USB CDCHID Device Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 16:50:42 --
-- : DBG key 1 2 used as buttons
-- : 1st press to push, 2nd press to release
-I- CDCDSerial_Initialize
-I- CDCDSerialPort_Initialize
-I- HIDDFunction_Initialize
USBD_Init
-I- VBus configuration
-I- conn
```

Typing “1” makes terminal print “a” continuously, while typing “1” again makes terminal stop outputting “a”.

```
-I- Key 0 pressed
aaaa
```

➢ **Procedures**

Download program into board, press NRSRT and observe relevant terminal information. When connect to PC by USB cable, the LED blinks and the host reports a new USB device attachment. After installing the driver, there will be “AT91 USB to Serial Converter” and “ATMEL Mass Storage MSD USB Device”. PC will add a new added 10M removable disk which can be used as a common disk.

➢ **Phenomenon Indicates**

After starting board and installing driver, the host will add a COM device. Refer to Figure 3-55:

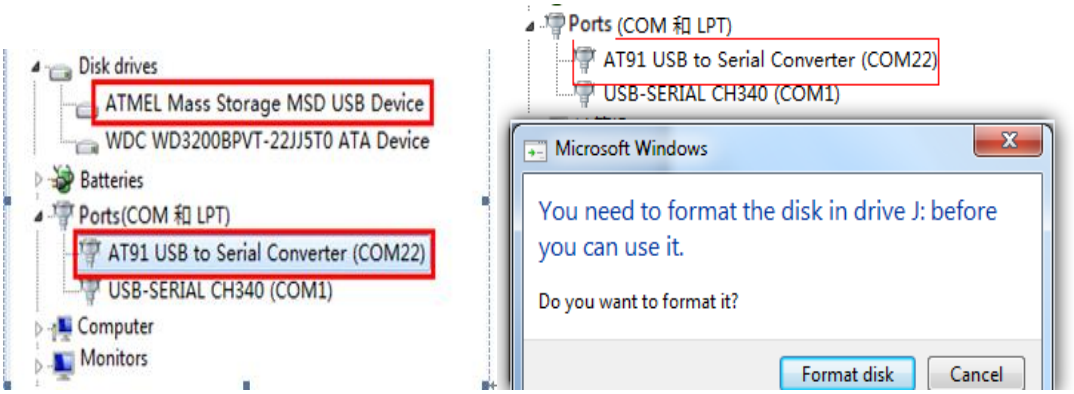


Figure 3-55

Terminal information:

```
-- USB CDCMSD Device Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 17:06:07 --
-I- LUN init
RamDisk @ 22000000, size 10485760
-I- RAM Disk init
-I- LUN init
-I- LUN: blkSize 1, size 20480
-I- CDCDSerial_Initialize
-I- CDCDSerialPort_Initialize
-I- MSDFun init
MSDReset USBD_Init
-I- VBus configuration
-I- connn
…
…
…
-----------------------------
```

```
TstUnitRdy Rdy Cplt
SendCSW ok
------------------------------
TstUnitRdy Rdy Cplt
SendCSW ok
------------------------------
TstUnitRdy Rdy Cplt
SendCSW ok
（More information is not a comprehensive display）
```

3.3.29 Usb_massstorage

➢ **Purpose**

This example demonstrates UDP as well as USB Framework that is used for USB drivers such as USB MSD.

➢ **Functional Description**

This example simulates a 10M bytes USB disk. When connect the board to the host by USB cable, the board works as a USB disk. If the board has SDRAM, the disk capacity can reach up to 10M, so that read/write speed can be tested. If there is no SDRAM but only internal flash, the disk is about 30K and only small file can be tested.

➢ **Procedures**

Download program into board, press NRSRT to observe relevant terminal information. When connect to PC by USB cable, the host reports a new USB device attachment and Disk installation. Then “ATMEL Mass Storage MSD USB Device” appears in hardware device list and the host pops up a dialog box that whether to format removable. As shown in Figure 3-52:

➢ **Phenomenon Indicates**

The device manger adds a new device. Refer to Figure 3-56:

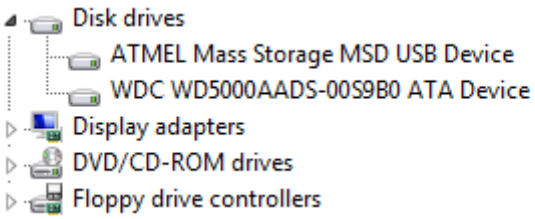


Figure 3-56

The host pops up a dialog box that whether to format the removable. Refer to Figure 3-57 and 3-58:

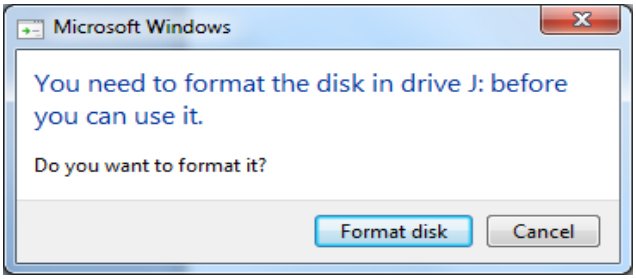


Figure 3-57

Figure 3-52 Before format

Figure 3-53 (Right) After format

Figure 3-58

Terminal information:

```
-- USB Device Mass Storage Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  3 2013 17:27:51 --
-I- LUN init
RamDisk @ 22000000, size 10485760
-I- RAM Disk init
-I- LUN init
-I- LUN: blkSize 1, size 20480
-I- MSDFun init
MSDReset USBD_Init
-I- VBus configuration
-I- conn
Rsm Susp Rsm NewReq Msdf Std gDesc Dev NewReq Msdf Std sAddr SetAddr(1)
NewReq Msdf Std gDesc Dev NewReq Msdf Std gDesc Cfg NewReq Msdf Std gDesc Str0
NewReq Msdf Std gDesc Str3 NewReq Msdf Std gDesc Cfg NewReq Msdf Std gDesc Str0
NewReq Msdf Std gDesc Str2 NewReq Msdf Std gDesc Str0 NewReq Msdf Std gDesc
Str2 NewReq Msdf Std gDesc Dev NewReq Msdf Std gDesc Cfg NewReq Msdf Std gDesc
Cfg NewReq Msdf Std gDesc Str0 NewReq Msdf Std gDesc Str0 NewReq Msdf Std gDesc
Str3 NewReq Msdf Std gDesc Str3 NewReq Msdf Std sCfg SetCfg(1) MSDFunCfg
```

```
MSDReset NewReq Msdf gMaxLun
    ------------------------------
    Inquiry Sending
    Inquiry Sent Cplt
    SendCSW ok
    ------------------------------
    -W- MSDD_ProcessCommand: Unknown cmd 0x23
    StaIN Cplt StallIn WNewReq Msdf ClrFeat Hlt Std cFeat Hlt aitHALT
    SendCSW ok
    …（Part is omitted）
```

3.3.30 RS485

- **Purpose**

  This example demonstrates how to use rs485 to communicate

- **Functional Description**

  This example communicates between two boards by rs485, demonstrates the use of rs485 communication. One board works as the sender, sends a string to another board continuously. The other board works as the receiver who receives and verifies the data and then displays the results.

  Procedures

  This example uses two MYD-SAM9X5-V2 development t boards. After download (two boards must download this routine program), connect J24 A1 and B1 of one board to their counterparts of another board. Then restart the development board, observe output information.

- **Phenomenon Indicates**

  Sending Terminal information:

```
RomBOOT
Start AT91Bootstrap...
Init DDR... Done!
Loading 1-Wire info...
sn: 0x4010424;      rev: 0x8421
Downloading image...
detected dataflash id = 0x1f 0x47 0x1 0x0 0x0.
Done!
```

```
-- RS485    Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    3 2013 17:43:31 --
---Press 1 It will be send the host-------
---Press 2 It will be receive the host----
-----It is sending ------
-----Send data:RS485 TEST OK!!!
```

Receiving Terminal information:

```
Start AT91Bootstrap...
Init DDR... Done!
Loading 1-Wire info...
sn: 0x4010424;      rev: 0x8421
Downloading image...
detected dataflash id = 0x1f 0x47 0x1 0x0 0x0.
Done!

-- RS485    Project 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug    3 2013 17:43:31 --
---Press 1 It will be send the host-------
---Press 2 It will be receive the host----
-----It is    receiving -------
-----Received Data:
RS485 TEST OK!!!
RS485 TEST OK!!!
RS485 TEST OK!!!
RS485 TEST OK!!!
RS485 TEST OK!!!
```

3.3.31 External GPIO

- **Purpose**

  This example describes how to use SPI to extend GPIO

- **Functional Description**

  This example controls SC16IS through SPI to produce general GPIO.

- **Procedures**

  Press NREST after downloading, and observe relevant terminal information. Output voltages in J17 are as follows: 1, 2, 5, 6 is 3.3 v. 3, 4, 7, 8 is 0 v.

- **Phenomenon Indicates**

Terminal information:

```
-- SPI with SC16 GPIO Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  9 2013 19:06:30 --
DMA driver initialized with IRQ
SPI and SC16 drivers initialized
Set SC16 GPIO output is 0xaa
Modify IOdir and IOstate to change PIN direction and state
```

3.3.32 External UART

- **Purpose**

  This example describes how to use SPI to extend UART.

- **Functional Description**

  This example controls SC16IS through SPI to produce the UART.

- **Procedures**

  Press NREST after downloading, and observe terminal information. Connect PC's serial port to J21, press some button to input, and observe echoed character.

- **Phenomenon Indicates**

  Terminal information:

```
-- SPI with SC16 Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  9 2013 20:50:02 --
DMA driver initialized with IRQ
SPI and SC16 drivers initialized
Set SC16 uart  baud rate 115200 and select SC16 channel B
Receive data is 1
Receive data is 1
Receive data is 1
Receive data is 1
Receive data is 1
Receive data is 1
```

  Serial Terminal information:

```
11111111111111111111111111
```

3.3.33 External RTC

- **Purpose**

  This example describes how to control clock chip by IIC.

- **Functional Description**

  The example controls reading and writing time of DS3231 by IIC.

- **Procedures**

  Download program into board, press NRSRT to observe relevant terminal information.

Press some button to input.

- **Phenomenon Indicates**

  Terminal information:

```
-- TWI DS3231 Example 2.0 --
-- SAM9XX5-EK
-- Compiled: Aug  9 2013 18:02:47 --
Configure buttons with debouncing.
Press K4 to read date.
Set date is : 8/8/13 4 23:59:30
Now date is: 8/9/13 5 0:0:4
Now date is: 8/9/13 5 0:0:7
Now date is: 8/9/13 5 0:0:11
```

Chapter 4 Linux System Guide

4.1 Outline

This Chapter describes how to run Linux system and embedded Linux applications, and the process of drive development in MYD-SAM9X5-V2 development board. It includes building the development environment, compiling the source code, image downloading, Linux applications & drivers anglicizing and Qt transplanting. The default startup is from the NandFlash and the initial system at factory is Linux system. The storage structure of NandFlash and some analysis are as follows:

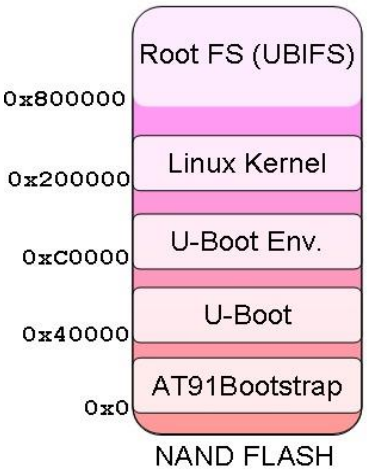


Figure 4-1

(1) Boot Strap

After power on, the first class boot program is copied automatically to internal SRAM and begins to be implemented by CPU. The main role is to initialize CPU and external RAM, and u-boot is copied from NandFlash to external RAM, and then jump to u-boot entry and start u-boot.

(2) U-boot

The secondary boot program which is used for kernel image update, load kernel and boot kernel.

(3) U-boot Env

Configure environment variables and provide u-boot running parameters, such as ip

address, start a command and kernel boot parameters.

(4) Linux Kernel

Build Linux 2.6.39 kernel for MYD-SAM9X5-V2

(5) Root FS

Angstrom-X11 GUI system file, Angstrom-Qt no desktop file system

4.2 Software Resources

Category	Name	Remark
Boot program	Boot Strap	First Boot Program
	U-boot	Secondary Boot Program
Linux kernel	Linux 2.6.39	Linux Kernel Only for MYD-SAM9X5-V2 Hardware
Device Drivers	USB The Host	USB The host driver supports the mode of OHCI and EHCI transmission
	USB Device	USB Device Driver (Gadget)
	Ethernet	Ethernet Driver
	MMC / SD	MMC/SD Card Driver
	NandFlash	NandFlash/Smart Media Driver
	TWI(I2C)	I2C Driver
	SPI	SPI Driver
	WM8731	WM8731 Driver
	LCD Controller	LCD Driver, support 4.3 inch, 7 inch,10.2 inch
	RTC	RTC Clock Driver
	Touch-Screen	4-wire Resistive Touch Screen Driver
	PWM	PWM (pulse width modulation) Driver
	UART	Serial Port Driver
	LED	LED Driver, including GPIO LED PWM LED driver
System Files	Angstrom-X11	X11 file system with a graphical port
	Angstrom-Qt	(No desktop file system of Qt library transplanted)

Table 4-1

4.3 Start Linux System

4.3.1 Install Download Tool

Install Atmel ISP download software SAM-BA (requires version 2.11 or above, the software location in the DVD is: 03-Tools / SAM-BA), Note to uninstall samba v2.10 and previous versions (SAM-BA software and development board USB drive) before installing. If you need two or more versions of SAM-BA coexist, then different versions of SAM-BA should use different USB ports on PC.

4.3.2 Connect the Board to PC

(1) Connect the board (J8) with PC through a micro USB cable, and switch the power to 12V.

(2)Turn SW1_1, SW1_2 OFF, and disconnect jumper JP8, then restart board (this order cannot be disturbed). Firstly, connect to board, it will prompt to install board driver. Then select SAM-BA installation directory which is shown in Figure 4-2:

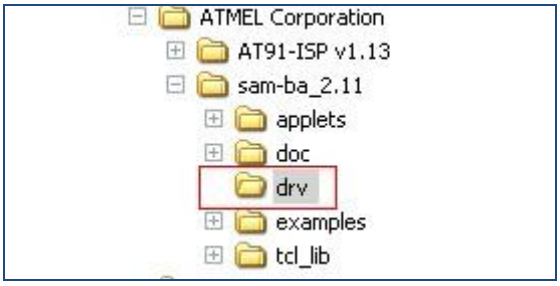


Figure 4-2

(3) If there is the same as Figure 4-3 in "my computer->properties->Management-> device manager-> port", it means that board driver has been installed.

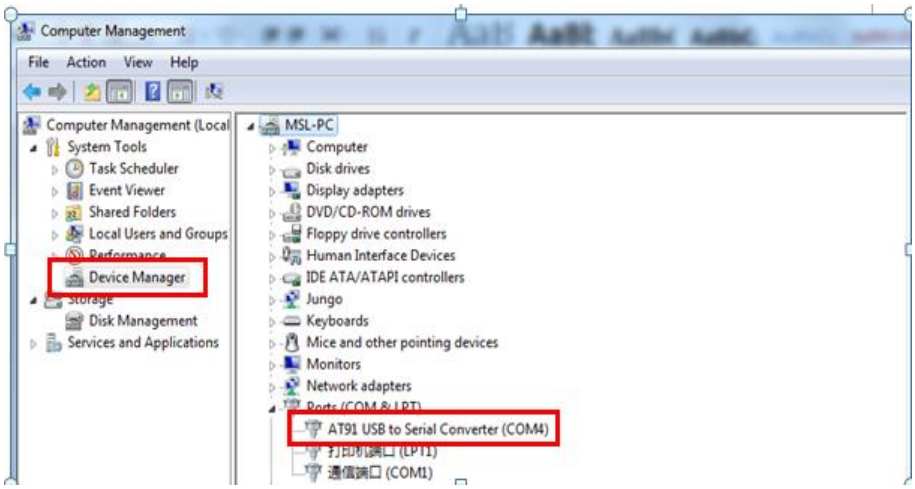


Figure 4-3

Here COM4 is the machine connection port (determined by actual situation, here is COM4). Switch SW1_1 is ON and switch SW1_2 is kept OFF.

(4) Connect USB_UART to PC by USB cable, set up HyperTerminal: COM1, 115200, 8, none, 1. COM port number is set by actual situation.

4.3.3 Automatic Download

Note: (1) Please pull out SD card before download, otherwise an error may occur.

(2) Here the example given is the X11 image for a 4.3-inch screen. If you are to use a different size screen or to download Qt image, please download the image under the corresponding directory.

After completing the steps in Chapter 4.3.1 and 4.3.2, open CD-ROM directory: “\02-Images\Linux\4.3LCD\X11”, double-click “at91sam9x5ek_demo_linux_nandflash.bat”. Then SAM-BA will download Linux image automatically to board. The entire download process takes about three minutes. When popping file “logfile.log” automatically, and reset board. There will display Linux start information

Please refer to Chapter 4.7 about how to use Linux.

4.3.4 Manual Download

Note: (1).please pull out SD card before download, otherwise an error may occur.

(2). Here the example given is the X11 image for a 4.3-inch screen. If you are to use

a different size screen or to download Qt image, please download the image under the corresponding directory.

Use SAM-BA to download Linux manually

(1) After completing the steps in Chapter 4.3.1 and 4.3.2, turn SW1_1, SW1_2 off, restart board, and then turn SW1_1 on. Open SAM-BA to set corresponding parameters. Therefore, connection is "\the USB serial\COMXX (XX is each computer's COM port, choose it by actual situation, here XX is COM8)", board select "at91sam9x35-ek". And then click "Connect", specific settings and connected results are shown in Figure 4-4, Figure 4-5:

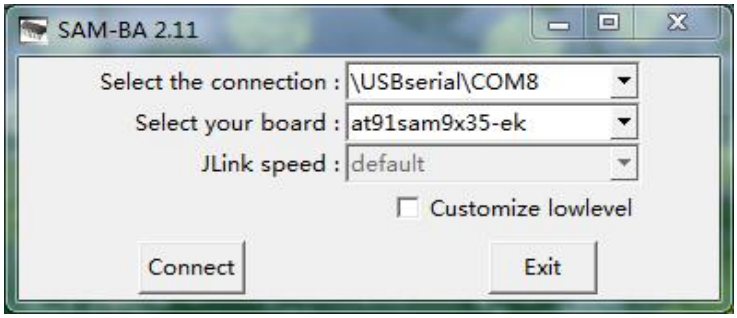


Figure 4-4

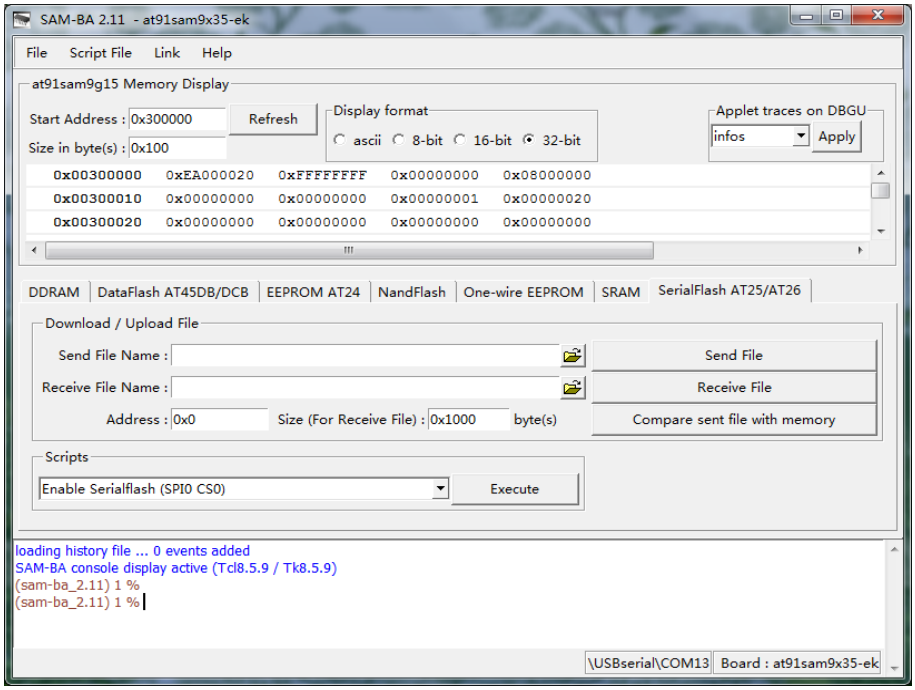


Figure 4-5

(2) Click NandFlash tab and execute "Enable NandFlash", "Enable OS PMECC parameters" and "Erase All" sequentially in "Scripts" tab. (click on nearby "execution" to

perform.). A setting popup dialog box will appear while executing “Enable OS PMECC”; click “OK” to use the default settings. The specific operations are as follows:

Select "Enable NandFlash" in Scripts tab, and then click "Execute" to Enable NandFlash. Refer to Figure 4-6:

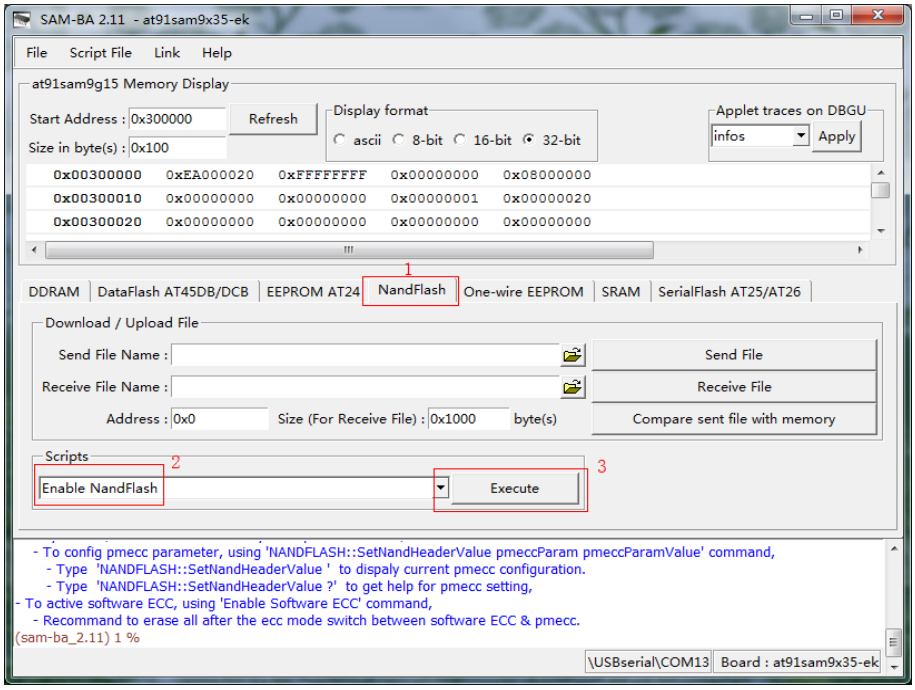

SAM-BA 2.11 - at91sam9x35-ek
File Script File Link Help
at91sam9g15 Memory Display
Start Address : 0x300000
Size in byte(s) : 0x100
Refresh
Display format
ascii 8-bit 16-bit 32-bit
Applet traces on DBGU
infos
Apply
0x00300000 0xEA000020 0xFFFFFFFF 0x00000000 0x08000000
0x00300010 0x00000000 0x00000000 0x00000001 0x00000020
0x00300020 0x00000000 0x00000000 0x00000000 0x00000000
1
DDRAM DataFlash AT45DB/DCB EEPROM AT24 NandFlash One-wire EEPROM SRAM SerialFlash AT25/AT26
Download / Upload File
Send File Name :
Receive File Name :
Address : 0x0
Size (For Receive File) : 0x1000 byte(s)
Send File
Receive File
Compare sent file with memory
Scripts
2
Enable NandFlash
Execute
3
- To config pmecc parameter, using 'NANDFLASH::SetNandHeaderValue pmeccParam pmeccParamValue' command,
- Type 'NANDFLASH::SetNandHeaderValue ' to dispaly current pmecc configuration.
- Type 'NANDFLASH::SetNandHeaderValue ?' to get help for pmecc setting,
- To active software ECC, using 'Enable Software ECC' command,
- Recommand to erase all after the ecc mode switch between software ECC & pmecc.
(sam-ba_2.11) 1 %
\USBserial\COM13 Board : at91sam9x35-ek

Figure 4-6

Select "Enable OS PMECC parameters" in Scripts tab and then click "Execute", click OK to use the default settings. Refer to Figure 4-7:

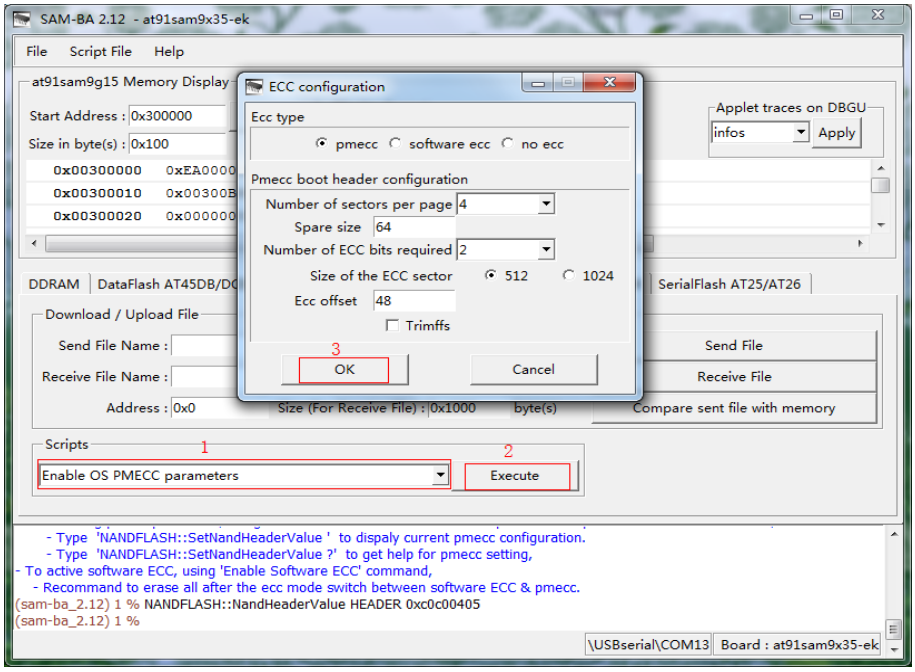

SAM-BA 2.12 - at91sam9x35-ek
File Script File Help
at91sam9g15 Memory Display
ECC configuration
Ecc type
pmecc software ecc no ecc
Pmecc boot header configuration
Number of sectors per page 4
Spare size 64
Number of ECC bits required 2
Size of the ECC sector 512 1024
Ecc offset 48
Trimffs
3
OK
Cancel
Applet traces on DBGU
infos
Apply
Download / Upload File
Send File Name :
Receive File Name :
Address : 0x0
Size (For Receive File) : 0x1000 byte(s)
Send File
Receive File
Compare sent file with memory
Scripts
1
Enable OS PMECC parameters
2
Execute
- Type 'NANDFLASH::SetNandHeaderValue ' to dispaly current pmecc configuration.
- Type 'NANDFLASH::SetNandHeaderValue ?' to get help for pmecc setting,
- To active software ECC, using 'Enable Software ECC' command,
- Recommand to erase all after the ecc mode switch between software ECC & pmecc.
(sam-ba_2.12) 1 % NANDFLASH::NandHeaderValue HEADER 0xc0c00405
(sam-ba_2.12) 1 %
\USBserial\COM13 Board : at91sam9x35-ek

Figure 4-7

Select "Erase All" in Scripts tab, then click "Execute", format NandFlash. Refer to Figure 4-8:

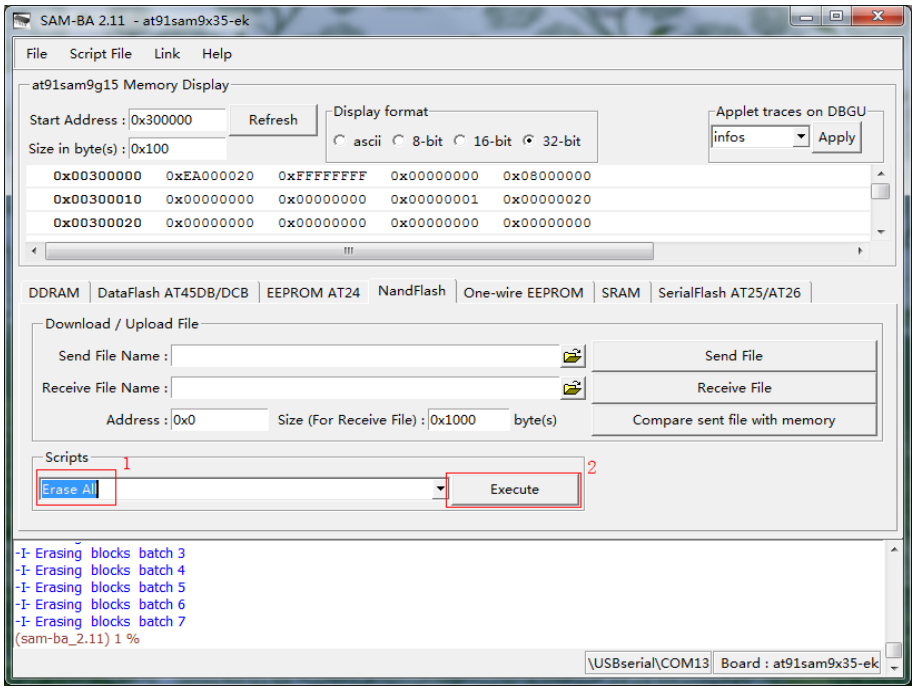


Figure 4-8

(3) Download “at91sam9x5ek-nandflashboot-3.1.bin”. Refer to Figure 4-9 and 4-10:

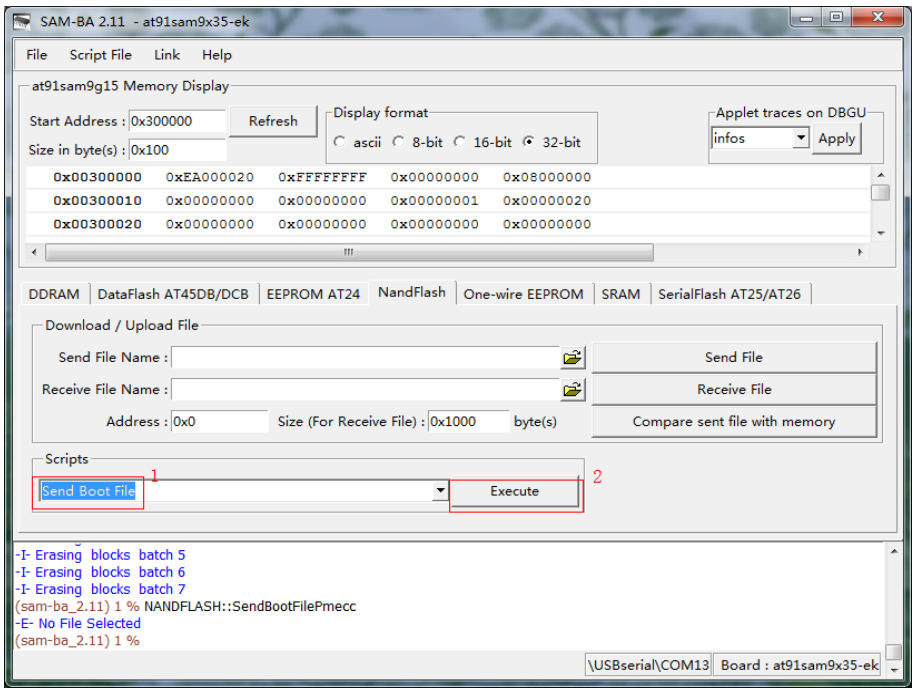


Figure 4-9

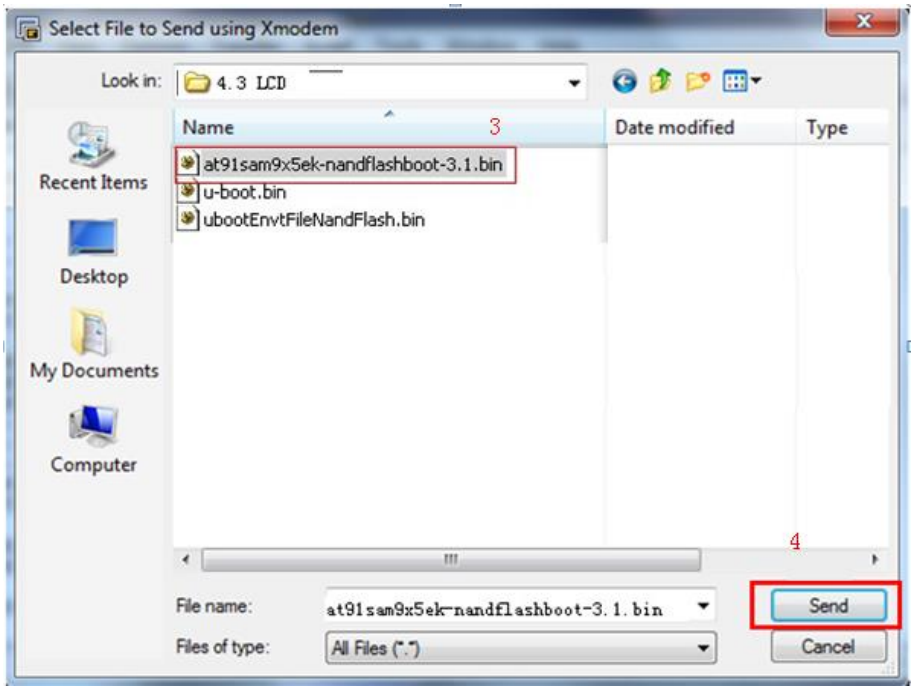


Figure 4-10

(4) Download “u-boot.bin” to 0x40000. Refer to Figure 4-11:

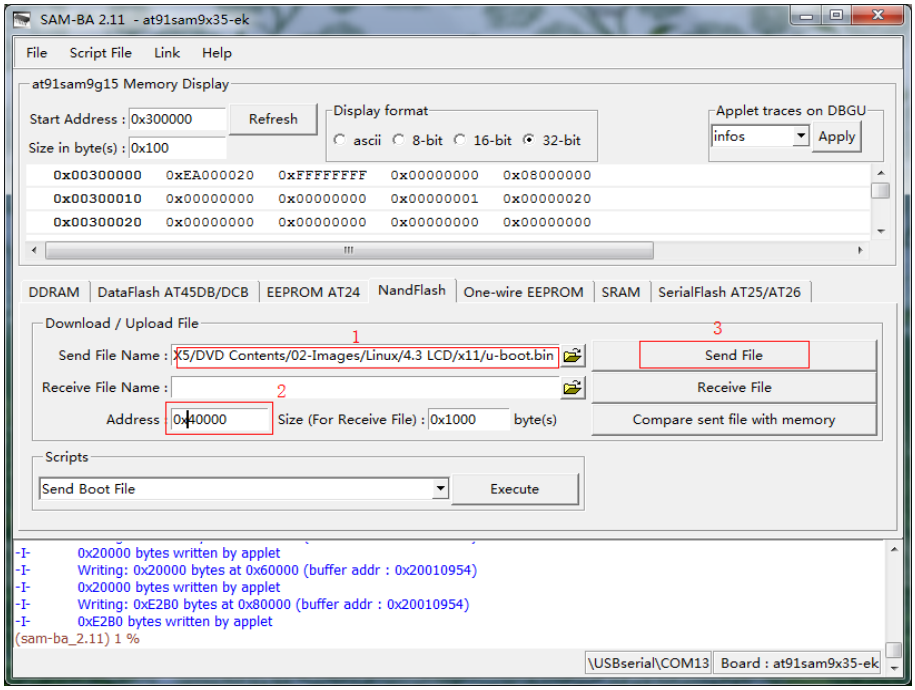


Figure 4-11

(5) Download u-boot “EnvtFileNandFlash.bin” to 0xC0000. Refer to Figure 4-12:

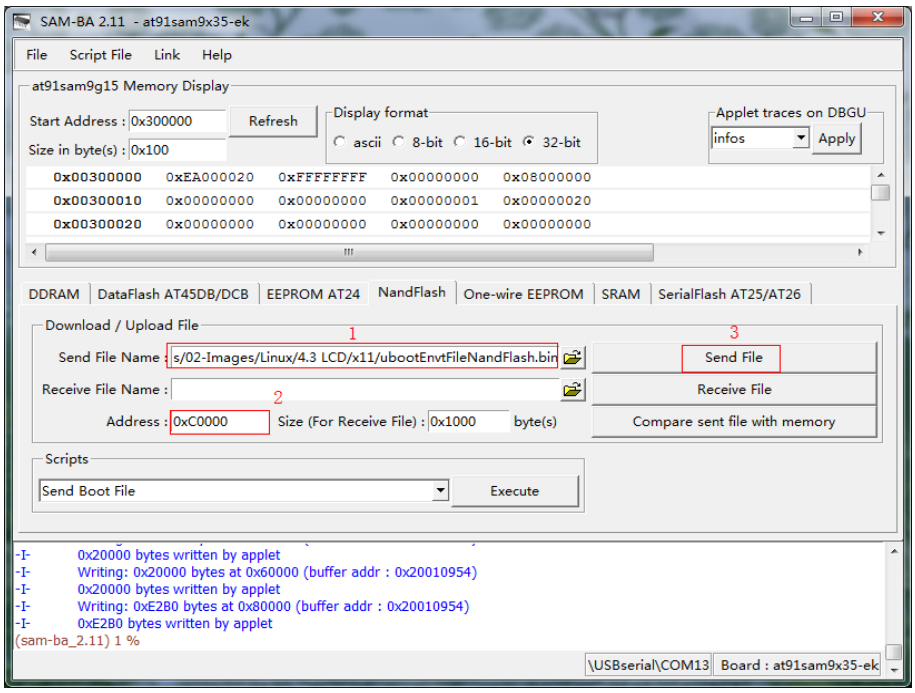


Figure 4-12

(6) Download Linux kernel Image to 0x200000. Refer to Figure 4-13:

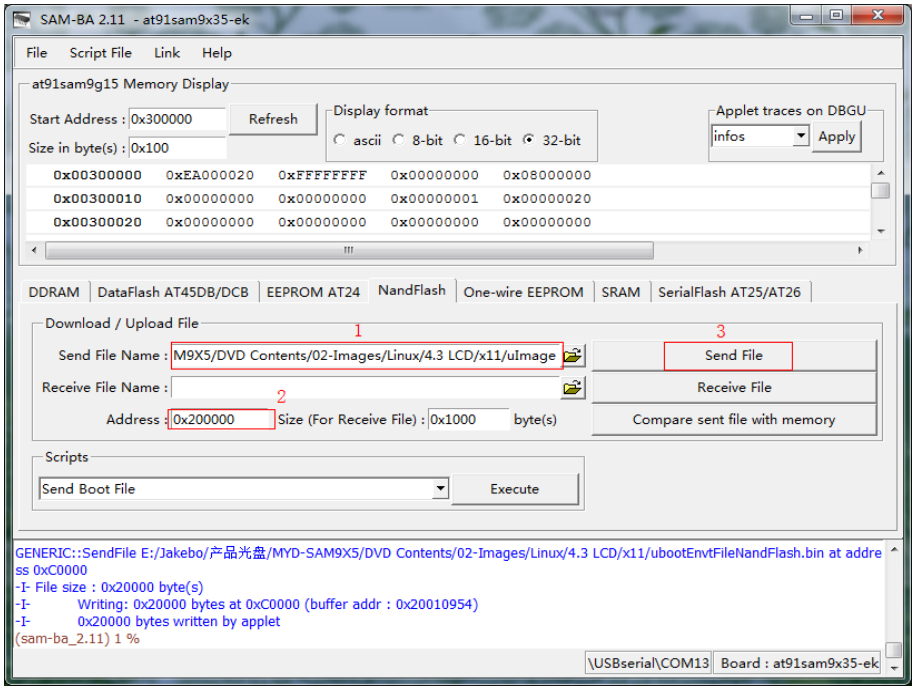


Figure 4-13

(7) Program system files

Download

“angstrom-x11-at91sam9-image-eglibc-ipk-v20110624-at91sam9x5ek.rootfs.ub” to 0x80000. Refer to Figure 4-14:

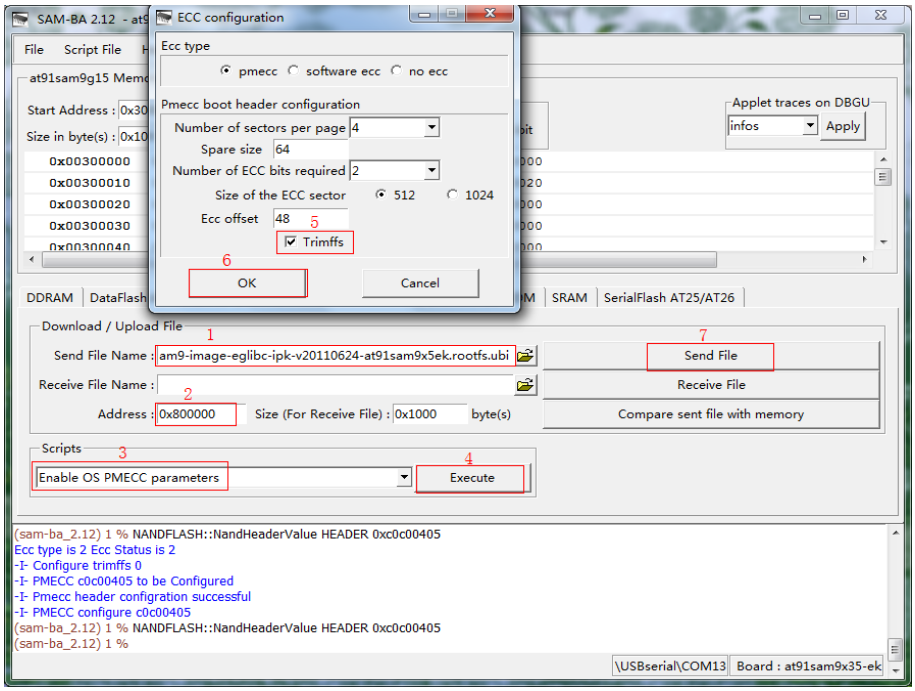


Figure 4-14

Note: Be sure to select “Trimffst” on the popup dialog box in the process of executing “Enable OS PMECC parameters”.

Finally, restart board to boot Linux system normally. Please refer to Chapter 4.7, about how to use Linux.

4.4 Linux Development Environment Structure

About the content of this Chapter, please refer to **"Description VirtualBox's Linux-based development environment to build. pdf"**.

4.5 Installation And Compile

4.5.1 Create a Working Directory

```
# mkdir /home/MYIR_SAM9X5
# cd /home/MYIR_SAM9X5
```

Copy 05-Linux_Source folder in CD to /home/MYIR_SAM9X5:

```
# cp -r /media/cdrom/05-Linux_Source ./
```

4.5.2 Install Cross Compiler Tools

Decompress cross compiler tool to /usr/local directory

```
# sudo tar xvjf   \
05-Linux_Source/CrossTool/   \
arm-2010q1-202-arm-none-linux-gnueabi.tar.bz2 -C   /usr/local
```

4.5.3 Install AT91Bootstrap Source and Compile

(1) Install

```
# tar xvjf 05-Linux_Source/AT91Bootstrap/AT91Bootstrap-5series_1.2.tar.bz2
-C ./
```

(2) Compile

```
# cd AT91Bootstrap-5series_1.2
# make distclean
# make at91sam9x5nf_defconfig
# make CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
```

“At91sam9x5ek-nandflashboot-3.1.bin” in binary directory is “AT91Bootstrap file”.

AT91Bootstrap is a boot loader for ATMEL chip, which initializes necessary hardware (GPIO Clock, SDRAM, etc.), then copies u-boot to SDRAM to run.

4.5.4 Install U-boot Source and Compile

(1) Install

```
# cd 05-Linux_Source/U-Boot/
# tar xvjf u-boot-2010.06.tar.bz2
# cd u-boot-2010.06
```

(2) Compile the u-boot without debug function

The “u-boot.bin” without debug function will not check the keys of key board when the development board starts, and will guide the system directly. By default, U-boot compiled without any modification has no debug function. The compiling command is as follows:

```
# make at91sam9x5ek_nandflash_config
# make CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
# ls
```

After compiling, the “u-boot.bin” without debug function will be generated under directory “U-boot compiled by default without any modification”

(3) Compile the u-boot with debug function

The U-boot operation mode uses “u-boot-debug.bin” to guide the system. It can be entered by pressing the Space/Enter key immediately after the development board startup. The board should be connected with the host through serial port. This mode can do tftp download, update the image, set u-boot environment variables, etc.. It will continue to guild the system if no key action is detected. As the u-boot with debug function will do a series of operations to initialize the network when the system starting, the startup speed will greatly slow down(ahout 10 to 15 seconds). So we strongly recommend to use the u-boot without debug function in actual product. Modifying the document of “/u-boot-linux/include/configs/at91sam9x5ek.h” is needed to open debug function.

① Modify CONFIG_BOOTDELAY to be 1 second in the 91th line. There will be 1 second’s waiting for input from the key board:

```
#define CONFIG_BOOTDELAY            0//1
```

Modify above to:

```
#define CONFIG_BOOTDELAY            1
```

② Cancle the note to CONFIG_MACB in the 168th line, thereby opening the network support:

```
//#define CONFIG_MACB                    1
```

Modify above to:

```
#define CONFIG_MACB                    1
```

③ Run the following command to compile:

```
# make    distclean
# make    at91sam9x5ek_nandflash_config
# make    CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
# ls
```

After compiling, the u-boot.bin with debug function will be generated under “u-boot-linux” directory.

4.5.5 Install and Compile Linux Kernel Source Code

(1) Install

```
# cd 05-Linux_Source/Linux_Kernel/
# tar xvjf linux-2.6.39.tar.bz2    -C    ./
# cd linux-2.6.39/
```

(2) Compile

Different sizes of LCD screen configuration files are as follows:

LCD size	Configuration file
4.3 inch	myir_MYD-SAM9X5_4.3lcd_defconfig
7.0 inch	myir_MYD-SAM9X5_7.0lcd_defconfig
10.2 inch	myir_MYD-SAM9X5_10.2lcd_defconfig

The auto (script) or manual mode can be used for compilation

① Compile “make_image.sh” script in source root directory directly.

```
# sudo chmod a+x make_image.sh
# ./make_image.sh 4.3      (4.3 means 4.3-inch screen)
```

② Or execute the following command to compile:

```
# make ARCH=arm myir_MYD-SAM9X5-V2_4.3lcd_defconfig
# make ARCH=arm uImage \
   CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
```

Note: The “make uImage” command requires that compile tool “mkimage” has been installed. Use the following command to install the tool.

Sudo apt-get install “U-boot-mkimage”:

After compiling the kernel, “uImage file” in "arch /arm/boot/” directory is Linux image file that we need.

4.6 Make Linux File System

The system file in product CD “Angstrom-x11-at91sam9-image-eglibc-ipk-v20110624-at91sam9x5ek.rootfs.ubi” can be custom-built. Here, take adding an application “helloworld” to the root directory of file system as an example to show the detailed steps of making file system.

4.6.1 Write Demo Program “helloworld”

Firstly, write a simple program “helloworld”:

(1) Generate and compile “helloworld.c”

```
# vi helloworld.c
```

Enter the following in the “helloworld.c”, save it and exit:

```
#include <stdio.h>
int main(int argc, char *argv[])
{
      int i;
```

```
        printf("========== Hello World ==========\n");
        printf("argc: %d\n", argc);
        for(i = 0; i < argc; i++)
        {
            printf("argv[%d]: %s\n", i, argv[i]);
        }
        return 0;
    }
```

(2) Compile “helloworld.c”

Add cross-compiler tools path to PATH:

```
# export PATH=$PATH:/usr/local/arm-2010q1/bin/
```

Use the cross compiler tool to compile:

```
# arm-none-linux-gnueabi-gcc -o helloworld helloworld.c
```

The application “helloworld” is generated

4.6.2 Mount UBIFS File System

UBIFS is a new flash file developed by NOLIA engineers under the help of Szeged University, which is considered as the next generation of JFFS2 files system. UBIFS file system was specifically designed for large-capacity FLASH embedded mobile devices. Mounting UBIFS file system must have mtd interface, while ordinary PC usually doesn’t utilize mtd to manage disk. So, here need nandsim simulator to simulate mtd device with a RAM space, then mount UBIFS file system. The concrete steps are as follows:

(1) Load UBIFS drive mtd driver

Enter the following command in turn:

```
# sudo modprobe nandsim first_id _byte=0xec second_id_byte=0xda
  third_id_byte=0x10 fourth_id_byte=0x95
```

It needs to pass a few parameters to load nandsim: first_id_byte, second_id_byte, third_id_byte and fourth_id_byte are ID for NANDFLASH of the simulated target. MYD-SAM9X5-V2 use NANDFLASH of Samsung K9F2G08U0B. Four ID bytes can be found in datasheet:

0xec,0xda,0x10,0x95

If executed above command successfully, there will be “mtd0” and “mtd0r0” devices in “/dev/directory”:

```
# ls /dev/mtd*
/dev/mtd0    /dev/mtd0ro
```

(2) Erase “mtd0” partition, the operation is as follows:

```
# sudo flash_eraseall /dev/mtd0
flash_eraseall has been replaced by `flash_erase <mtddev> 0 0`; please use it
Erasing 128 Kibyte @ ffe0000 -- 100 % complete
```

(3) Load “Angstrom-x11-at91sam9-image-eglibc-ipk-v20110624-at91sam9x5ek.rootfs.ubi” to new “mtd0” partition. Here used “dd” command to load as follows:

```
# sudo dd \
if=Angstrom-x11-at91samg-image-eglibc-ipk-v20110624-at91sam9x5ek.rootfs.ubi \
of=/dev/mtd0
100864+0 records in
100864+0 records out
51642368 bytes (52 MB) copied, 0.323121 s, 160 MB/s
```

(4) Mount UBIFS system file

After completing the above step, load “ubi” modules and attach it to “mtd0”, then mount UBIFS file system like mounting ordinary mtd device.

Load ubi modules and attach it to “mtd0” equipment:

```
# sudo modprobe ubi mtd=0,2048
```

Create a new mounting point:

```
# mkdir fsmount
```

Mount it by the following command:

```
# sudo mount -t ubifs ubi0_0 fsmount/
# ls fsmount/
bin boot dev etc home lib media mnt proc sbin sys tmp usr va
```

Now that mounting UBIFS file is successful.

4.6.3 Modify UBIFS System Files

After mounting UBIFS file system successfully, we can modify file content on-demand, such as adding deleting or modifying the files. The example describes adding complied demo program “helloworld” to system root directory. The operation is as follows:

```
# sudo cp helloworld fsmount/
# sync
# ls fsmount
bin boot dev etc helloworld home lib media mnt proc sbin sys tmp usr var
```

4.6.4 Regenerate UBIFS System File

After modification, it needs to regenerate the file system by “mkfs.ubifs” tool. If you haven’t installed the “mkfs.ubifs” tools, using the following command to install using:

```
# sudo apt-get install mtd-utils
```

Enter the following command to generate a new UBIFS file system:

```
# sudo mkfs.ubifs -r fsmount/ -m 2048 -e 126976 -c 2024 \
  -o ubifs.img
```

mkfs.ubifs Parameter Description:

-r Establish the system file directory
-m Minimum I/O transfer unit size
-e Logical size of erase block
-c The largest number of erase logic blocks
-o Specify the output file

View the smallest I/O transfer unit size and logical erase block size by the following command:

```
# ubinfo /dev/ubi0
ubi0
Volumes count:                           1
Logical eraseblock size:                 126976 bytes, 124.0 KiB
Total amount of logical eraseblocks:     2048 (260046848 bytes, 248.0 MiB)
Amount of available logical eraseblocks: 0 (0 bytes)
Maximum count of volumes                 128
Count of bad physical eraseblocks:       0
Count of reserved physical eraseblocks:  20
Current maximum erase counter value:     1
Minimum input/output unit size:          2048 bytes
Character device major/minor:            250:0
Present volumes:                         0
```

View erase block number by using the following command:

```
# ubinfo /dev/ubi0_0
Volume ID:   0 (on ubi0)
Type:        dynamic
Alignment:   1
Size:        2024 LEBs (256999424 bytes, 245.1 MiB)
State:       OK
Name:        rootfs
Character device major/minor: 250:1
```

Then use “ubinize” tool to generate “fsimage.ubi” file. Firstly, we need to create configuration files “ubinize ubinize.cfg”:

```
# vi ubinize.cfg
```

Enter the following text, save the file and exit:

```
[ubifs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=64MiB
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
vol_alignment=1
```

Enter the following command to generate the final “fsimage.ubi” file:

```
# sudo ubinize -m 2048 -p 128KiB -o fsimage.ubi ubinize.cfg
```

Parameter description of ubinize:

-m Size of minimum input/output byte flash unit
-p Erase block size of FLASH physical
-o output file

Here is different from the “mkfs.ubifs” parameters, “-p” parameter represents the physical erase block size. The UBI works in the MTD layer, so it needs the MTD parameters, namely physical parameters. The UBIFS works in the UBI, so it needs the UBI parameters, namely the logic parameter. Now, UBI image has been saved in the “ubi.img”, which not only contains UBIFS information, but also contains UBI information.

After completion, we can use the method described in Chapter 4.3.4 to download the generated file “fsimage.ubi” to 0x800000.

Reset board and input root to login, there is the newly added file “helloworld” in the root directory:

```
at91sam9x5ek login: root
root@at91sam9x5ek:~# cd /
root@at91sam9x5ek:/# ls
bin          etc          lib          proc         tmp
boot         helloworld   media        sbin         usr
dev          home         mnt          sys          var
```

Run “helloword” using the following command:

```
root@at91sam9x5ek:/# ./helloword
========== Hello World ==========
argc: 1
argv[0]: ./helloword
```

4.7 Linux Application

After running Linux system, it can be operated by touch screen operation or carried out by terminal serial. In the following Chapter, we exemplify how to operate Linux, how to mount U-disk/SD card, how to test network port and how to play music.

4.7.1 Touch Screen Calibration

Note: MYD-SAM9X25-V2 and MYD-SAM9G25-V2 board don't support touch screen.

On the first time logging in, the system will calibrate the screen using the default calibration parameters. If there is still a deviation after calibration, recalibrate the touch screen by the following steps:

(1) Open HyperTerminal (baud rate: 115200 Data bits: 8, Parity: None Stop bits: 1, data flow control: none). After starting Linux, log in as root command:

```
at91sam9x5ek login: root
```

(2) Run the calibration procedure and click the five corresponding calibration points on the LCD screen. The calibration can be carried out:

```
root@at91sam9x5ek:~# ts_calibrate
xres = 480, yres = 272
Took 33 samples...
Top left : X =  804 Y =  178
Took 40 samples...
Top right : X =  790 Y =  953
Took 31 samples...
Bot right : X =  301 Y =  950
Took 34 samples...
Bot left : X =  306 Y =  172
Took 30 samples...
Center : X =  550 Y =  562
-33.023254 -0.004476 0.489318
330.122131 -0.348463 -0.004259
Calibration constants: -2164212 -293 32067 21634884 -22836 -279 65536
```

(3) After the calibration is completed, restart the system to take effect. The operations are as follows:

```
root@at91sam9x5ek:~# sync
root@at91sam9x5ek:~# reboot
```

4.7.2 U-disk Use

(1) After entering Linux system by terminal, you can see the following information in HyperTerminal if the U-disk is connected through either USB host port of J9 or J10.

```
scsi0 : usb-storage 1-2:1.0
scsi 0:0:0:0: Direct-Access Kingston DT 101 G2 PMAP PQ: 0 ANSI: 0 CCS
sd 0:0:0:0: [sda] 7669824 512-byte logical blocks: (3.92 GB/3.65 GiB)
sd 0:0:0:0: [sda] Write Protect is off
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Assuming drive cache: write through
sda: detected capacity change from 0 to 3926949888
sda: sda4
sd 0:0:0:0: [sda] Assuming drive cache: write through
sd 0:0:0:0: [sda] Attached SCSI removable disk
FAT: invalid media value (0x01)
VFS: Can't find a valid FAT filesystem on dev sda.
EXT2-fs (sda): error: can't find an ext2 filesystem on dev sda.
FAT: invalid media value (0x01)
VFS: Can't find a valid FAT filesystem on dev sda.
```

Note: The above orange text error message shows a failure about the system attempting to mount a USB device to “sda”. This error can be ignored, because “sda” is not a valid partition. Above red text “sda4” is an effective partition that we mount.

(2) The system will mount a connected U-disk automatically, entering the following command to view the content of U-disk.

```
root@at91sam9x5ek:/# ls /media/sda4/
9x5??.rar          GTHE HOSTXP.GHO      NTLDR              helloworld
????????.TXT       MYD-S5PV210          PETOOLS            rootfs.tar
BOOT               NTDETECT.COM         WXPE               sam-ba_2.11.exe
```

(3) Unplug the U-disk directly after completing usage, system will uninstall it automatically.

4.7.3 SD Card Use

(1) Plug Micro-SD card into a Micro-SD card slot, and system will mount it automatically.

(2) When Micro-SD card is inserted, HyperTerminal displays SD card information:

```
mmc0: the host does not support reading read-only switch. assuming write-enable.
```

```
mmc0: new high speed SD card at address e624
mmcblk0: mmc0:e624 SU02G 1.84 GB
mmcblk0: detected capacity change from 0 to 1977614336
mmcblk0: retrying using single block read
```

(3) View SD card content:

```
root@at91sam9x5ek:/# ls /media/mmcblk0/
MyHeartWillGoOn.wav
```

(4) Pull out SD card directly, system will uninstall it automatically.

4.7.4 Play MP3 Music

Before playing music, connect headphones or stereo to J7. Storage an mp3 music in a U-disk and insert it into a USB port.

Play music in U-disk by mplayer command in terminal:

```
root@at91sam9x5ek:/media/sda4# mplayer thelastone.mp3
```

At this point, you can hear music from headphones. Terminal displays information as below, enter “Ctrl+C” to end playing music:

```
MPlayer UNKNOWN-4.5.3 (C) 2000-2010 MPlayer Team
Playing thelastone.mp3.
Alignment trap: mplayer (1091) PC=0x002560a0 Instr=0xe1d130b0 Address=0x40a1c455 FSR 0x001
Alignment trap: mplayer (1091) PC=0x002560a0 Instr=0xe1d130b0 Address=0x40a1c455 FSR 0x001
Audio only file format detected.
Clip info:
 Title:
 Artist:
 Album:
 Year:
 Comment:
 Genre: Unknown
==========================================================
Forced audio codec: mad
Opening audio decoder: [libmad] libmad mpeg audio decoder
AUDIO: 44100 Hz, 2 ch, s16le, 224.0 kbit/15.87% (ratio: 28000->176400)
Selected audio codec: [mad] afm: libmad (libMAD MPEG layer 1-2-3)
==========================================================
AO: [alsa] 48000Hz 2ch s16le (2 bytes per sample)
Video: no video
Starting playback...
```

```
A:     1.5 (01.4) of 239.0 (03:59.0) 19.7%
```

4.7.5 Network Port Test

Note: MYD-SARM9G15 dones't support network, MYD-SAM9G25/G35/X35 supports one network port connected J11 to use. MYD - SAM9X25 supports two Ethernet ports connect J11 or J10 to use

Connect the board to PC by a crosswire network cable or with a switch or router by a straight-through cable. Please note: On the method of connecting the board to PC using a crosswire cable, if board has extranet access demand, it requires the PC has dual network cards. Connect network card 1 to board, network card 2 to network. And the network card1 and 2 are set to bridge pattern. The following test is the mode of crosswire and PC Dual LAN bridges.

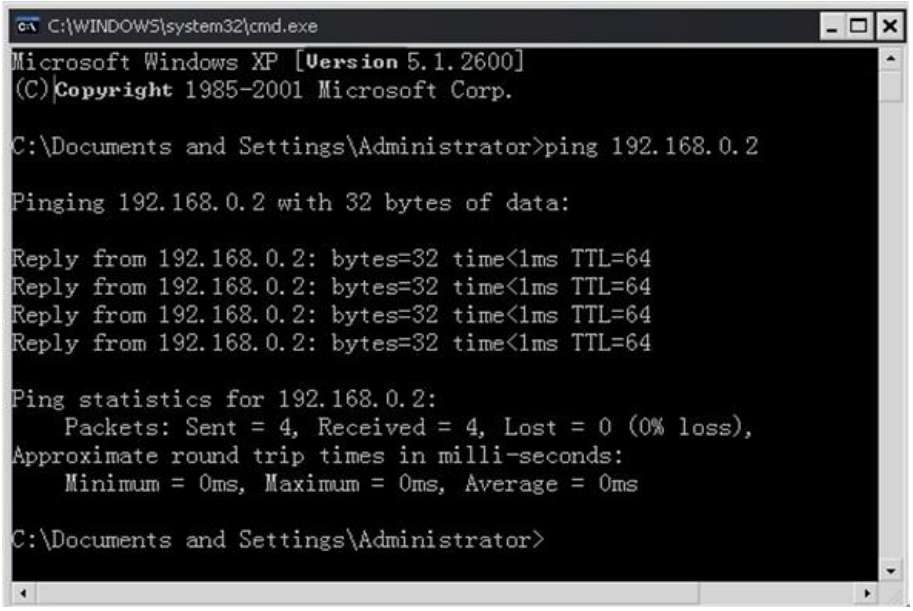
(1) In "Network Connections" window, select two network adapters, right-click and select "bridge" to bridge two network cards.

(2) Assign to board a IP, which is not occupied by any other devices of current LAN. In this case,we use address: 192.168.0.2:

```
root@at91sam9x5ek:/# ifconfig   eth0   192.168.0.2   up
```

(3) Test the connection between PC and the board by ping command (here the host IP: 192.168.0.3).

Ping board:


C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C)Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>ping 192.168.0.2
Pinging 192.168.0.2 with 32 bytes of data:
Reply from 192.168.0.2: bytes=32 time<1ms TTL=64
Reply from 192.168.0.2: bytes=32 time<1ms TTL=64
Reply from 192.168.0.2: bytes=32 time<1ms TTL=64
Reply from 192.168.0.2: bytes=32 time<1ms TTL=64
Ping statistics for 192.168.0.2:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\Administrator>

Ping board:

```
root@at91sam9x5ek:~# ping 192.168.0.3
PING 192.168.0.3 (192.168.0.3): 56 data bytes
64 bytes from 192.168.0.3: seq=0 ttl=64 time=3.767 ms
64 bytes from 192.168.0.3: seq=1 ttl=64 time=0.458 ms
```

4.7.6 Telnet Test

(1) Configure IP address

```
root@at91sam9x5ek:/# ifconfig eth0 192.168.0.2 up
```

(2) Configure Gateway

Test connection with the gateway, as follows:

```
root@at91sam9x5ek:/ # ping 192.168.0.1
PING 192.168.0.1 (192.168.0.1): 56 data bytes
64 bytes from 192.168.0.1: seq=0 ttl=64 time=5.776 ms
64 bytes from 192.168.0.1: seq=1 ttl=64 time=0.484 ms
```

Set 192.168.0.1 as the default gateway:

```
root@at91sam9x5ek:/ # route  add  default  gw  192.168.0.1
```

Test connection with 202.112.17.137:

```
root@at91sam9x5ek:/ # ping  202.112.17.137
PING 202.112.17.137 (202.112.17.137): 56 data bytes
64 bytes from 202.112.17.137: seq=0 ttl=52 time=26.592 ms
64 bytes from 202.112.17.137: seq=1 ttl=52 time=25.140 ms
```

(3) Use telnet to access BBS forum:

```
root@at91sam9x5ek:/media# telnet 202.112.17.137
```

Telnet 202.112.17.137
华南木棉BBS 最近 (1,10,15) 分钟平均负荷为 0.00 0.00 0.00 [负荷正常]
欢 迎 莅 临
中国教育和科研计算机网(CERNET)华南地区网络中心
电子公告牌华南网木棉站
本站地址: bbs.gznet.edu.cn (202.112.17.137)
Warmly Welcome to Bulletin Board Service(BBS) of
CERNET Southern Regional Center
If you have any problems, please send email to
scutbbs@scut.edu.cn
请用户遵守国家法律和CERNET用户守则, 谢谢合作!
欢迎光临【 华南木棉BBS 】 [Add '.' after YourID to login for BIG5]
目前上站人数: [2/25000]。 最高人数记录: [12970]。
请输入帐号(试用请输入 'guest'):
半:

It shows telnet test is successful.

(4) Configure DNS server

View current DNS server address by “ipconfig/all” command, machine DNS is 202.103.24.68. Set target board's DNS (depend on the circumstances):

```
root@at91sam9x5ek:/# echo "nameserver 202.103.24.68" | tee /etc/resolv.conf
```

Ping www.baidu.com to test extranet access:

```
root@at91sam9x5ek:/# ping www.baidu.com
PING 119.75.217.56 (119.75.217.56): 56 data bytes
64 bytes from 119.75.217.56: seq=0 ttl=54 time=60.990 ms
64 bytes from 119.75.217.56: seq=1 ttl=54 time=59.644 ms
```

Access extranet successfully.

4.7.7 Login Development Board By Telnet

(1) Configure IP (IP can be allocated, in case of IP 192.168.1.166)

```
# ifconfig eth0 192.168.1.166
```

(2) Start telenet service

```
# telnetd
```

(3) Login board by telenet client

Here use telenet in windows as example, input command as follow:

```
telnet 192.168.1.166
```

Then enter username “root” and press enter, login to development board through Telnet, as shown in Figure 4-15:

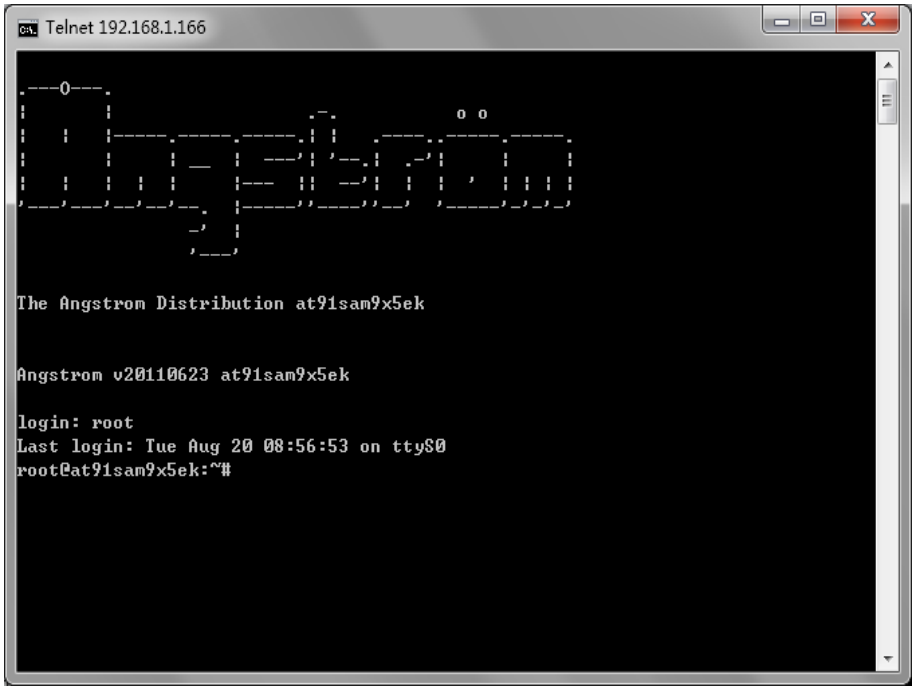


Figure 4-15

4.7.8 Mount NFS File System

Note: here use building NFS server on Ubuntu as an example.

(1) Build NFS server on Ubuntu

① install nfs-kernel-server

```
# sudo apt-get install nfs-kernel-server
```

② Add the following content at the end of file “/etc/exports” (The NFS directory is /home/jbo/ nfsdir this directory will be mount on clients)

```
/home/jbo/nfsdir *(rw,sync,no_root_squash)
```

③ Restart portmap and nfs-kernel-server

```
# service portmap restart
# service nfs-kernel-server restart
```

(2) mount NFS file system on board

① Configure IP

```
# ifconfig eth0 192.168.1.166
```

② Make directory

```
# mkdir nfs
```

③ Mount NFS file system, the NFS server IP is 192.168.1.223

```
# mount -t nfs -o nolock 192.168.1.223:/home/jbo/nfsdir nfs
```

Afte mount, NFS directory can be manipulated. These actions will be reflected to "/home/jbo/nfsdir" directly.

4.7.9 RTC Use

(1) Install button battery to board.

(2) System will set a default time at the first startup, so it needs to set the time after system startup.

Note: due to X11 system starts Scheduled Tasks service (atd), writing to hardware directly by "hwclock" command. The RTC will prompt an error that the device is busy. You can stop atd service and start if after test.

Stop atd Service:

```
root@at91sam9x5ek:/# /etc/init.d/atd stop
```

Set system time:

```
root@at91sam9x5ek:/# date -s 2012.07.08-10:36:00 ; hwclock -w
Sun Jul  8 10:36:00 BST 2012
```

Query system time and hardware RTC time:

```
root@at91sam9x5ek:/# date
Sun Jul  8 10:37:13 BST 2012
root@at91sam9x5ek:/# hwclock -r
Sun Jul  8 10:36:22 2012  0.000000 seconds
```

Recovery atd service:

```
root@at91sam9x5ek:/# /etc/init.d/atd start
```

4.7.10 Serial Port

MYD-SAM9X5-V2 can support up to 7 serial ports: a USB to serial port, two 3-lines UART, four 5-lines UARTs (including two extend 5-lines UART). In Linux system, SAM9G

15, SAM9G35, SAM9X35 have been registered five default serial ports; SAM9G25, SAM9X25 have been registered six serial ports (include USART3), and JP6 jumper must be connected. In Linux system, serial ports and their corresponding “Device node” and “Functional Description” are shown as shown in Table 4-2:

serial Port	Device Node	Functional Description
USB_UART	/dev/ttyS0	Debug UART
USART0(J18)	/dev/ttyS1	5-lines serial or RS485, Switch through the JP4 jumper
USART3(J19)	/dev/ttyS4	5-lines serial or RS485, Switch through JP4 jumper(only 9g25/9x25 have, when using JP6 jumper must be connected)
UART0(J22)	/dev/ttyS5	3-lines serial
UART1(J22)	/dev/ttyS6	3-lines serial
External Serial Port0(J20)	/dev/spi_usart0	5-lines serial
External Serial Port 1(J21)	/dev/spi_usart1	5-lines serial

Table 4-2

4.7.11 External GPIO

MYD-SAM9X5-V2 extends eight GPIO by J17. Directory is “/sys/class/GPIO”. Here take GPIO5 for an example.

(1) Export GPIO5:

```
root@at91sam9x5ek:/# cd /sys/class/gpio
root@at91sam9x5ek:/sys/class/gpio# echo 205 > export
```

External GPIO Base is 200, GPIO export number is 200 plus GPIO number, so exported GPIO5 is 205. After performed above command, there will generate “spi -gpio205” in current directory.

(2) Set GPIO to input/output

① Set GPIO to output:

```
root@at91sam9x5ek:/sys/class/gpio# cd spi-gpio205
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# echo out > direction
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# echo 1 > value
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# cat value
1
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# echo 0 > value
```

```
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# cat value
0
```

When setting the value, test GPIO by multi-meter.

② Set GPIO to input:

```
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# echo in > direction
```

If current GPIO is low level, the value is 0:

```
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# cat value
0
```

If current GPIO is high level, the value is 1:

```
root@at91sam9x5ek:/sys/class/gpio/spi-gpio205# cat value
1
```

4.8 Linux Driver Development Example

This Chapter describes a simple character device driver development achieving the function to control LEDs.

4.8.1 Hardware Schematic

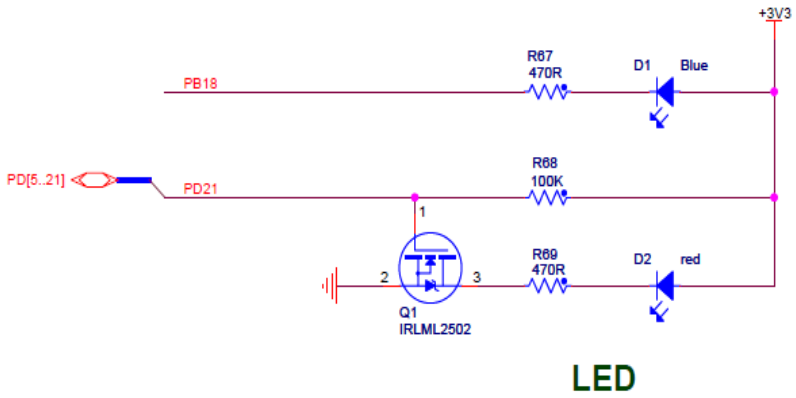


Figure 4-16

Use PD21 port to control D2 by IRLML2502. When it is high, LED turns on. Port PB18 controls D1 directly. When it is low, LED turns on. Refer to Figure 4-16:

4.8.2 Driver Source Code

(1) Create driver file in new kernel

Create a driver source file in directory “driver/char/”:

```
# cd linux-2.6.39
# vi    drivers/char/ledtest.c
```

(2) Driver source code “ledtest.c” is as follows：

```
#include <linux/string.h>
#include <linux/cdev.h>
#include <linux/fs.h>
#include <mach/gpio.h>
#include <linux/device.h>

#define DEVICE_NAME       "MYS-SAM9X5-ledtest"
static int LED_Major = 0;

struct cdev cdev;

#define    LED_OFF          0
#define    LED_ON           1

static unsigned long led_Table [] =
{
      AT91_PIN_PB18,    /**led_blue**/
      AT91_PIN_PD21,    /**led_red**/
};

static int MYS_SAM9X5_ledtest_open(struct inode *inode, struct file *file)
{
      printk("MYS-SAM9X5-ledtest Driver Open Called!\n");
      return 0;
}

static long MYS_SAM9X5_ledtest_ioctl(struct file *filp, unsigned int cmd, unsigned long
arg)
{
      if((cmd != 1 && cmd != 0) || (arg != 1 && arg != 0))
            return -1;

      switch(cmd)
      {
            case LED_ON:
                  if(arg)
                  {
                        at91_set_gpio_value(led_Table[arg], 1);
                  }
                  else
                  {
```

```
                    at91_set_gpio_value(led_Table[arg], 0);
                }
                break;
            case LED_OFF:
                if(arg)
                {
                    at91_set_gpio_value(led_Table[arg], 0);
                }
                else
                {
                    at91_set_gpio_value(led_Table[arg], 1);
                }
                break;
            default:
                return -EINVAL;
        }
        return 0;
}

static int MYS_SAM9X5_ledtest_release(struct inode *inode, struct file *file)
{
        printk("MYS_SAM9X5_LED Driver Release Called!\n");
        return 0;
}

static struct file_operations MYS_SAM9X5_ledtest_fops =
{
        .owner              =   THIS_MODULE,
        .open               =   MYS_SAM9X5_ledtest_open,
        .release            =   MYS_SAM9X5_ledtest_release,
        .unlocked_ioctl     =   MYS_SAM9X5_ledtest_ioctl,
};

static struct class *MYS_SAM9X5_ledtest_class = NULL;

static int __init MYS_SAM9X5_ledtest_init(void)
{
        int result,err;
        dev_t devno   =   MKDEV(LED_Major, 0);
        if (LED_Major)
        {
            result = register_chrdev_region(devno, 1, DEVICE_NAME);
            printk("Got the Major number by register_chrdev_region !\n ");
        }
```

```
    else
    {
        result = alloc_chrdev_region(&devno, 0, 1, DEVICE_NAME);
        LED_Major=MAJOR(devno);
        printk("Got the Major number by alloc_chrdev_region !\n");
    }

    if (result < 0)
    {
        printk(DEVICE_NAME " can't register major number\n");
        return result;
    }

    printk("register MYS_SAM9X5_ledtest Driver OK! Major = %d\n", LED_Major);
    cdev_init(&cdev,&MYS_SAM9X5_ledtest_fops);
    cdev.owner=THIS_MODULE;
    cdev.ops=&MYS_SAM9X5_ledtest_fops;
    err=cdev_add(&cdev, MKDEV(LED_Major, 0), 1);
    if (err)
    {
        printk("error %d adding led \n ", err);
        goto fail_cdev_add;
    }
    MYS_SAM9X5_ledtest_class = class_create(THIS_MODULE, DEVICE_NAME);
    if(IS_ERR(MYS_SAM9X5_ledtest_class))
    {
        printk("Err: failed in MYS_SAM9X5_ledtest class. \n");
        goto fail_create_class;
    }
    device_create(MYS_SAM9X5_ledtest_class, NULL, MKDEV(LED_Major, 0), NULL,
DEVICE_NAME);
    at91_set_gpio_output(AT91_PIN_PB18, 1);
    at91_set_gpio_output(AT91_PIN_PD21, 1);

    at91_set_deglitch(AT91_PIN_PB18, 1);
    at91_set_deglitch(AT91_PIN_PD21, 1);

    printk(DEVICE_NAME " initialized\n");

    return 0;

    fail_create_class:
    cdev_del(&cdev);
    fail_cdev_add:
```

```
    unregister_chrdev_region(devno, 1);

    return -1;
}
static void __exit MYS_SAM9X5_ledtest_exit(void)
{
    printk("MYS_SAM9X5 LED DRIVER MODULE EXIT\n");
    device_destroy(MYS_SAM9X5_ledtest_class, MKDEV(LED_Major, 0));
    class_destroy(MYS_SAM9X5_ledtest_class);
    cdev_del(&cdev);
    unregister_chrdev(LED_Major, DEVICE_NAME);
}
module_init(MYS_SAM9X5_ledtest_init);
module_exit(MYS_SAM9X5_ledtest_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Alvin");
MODULE_DESCRIPTION("This is an example of MYS_SAM9X5_LEDTEST drivers");
MODULE_ALIAS("A simplest module.");
```

4.8.3 Compile Driver

(1) Create a Makefile.

Create a “Makefile” on “ledtest”:

```
# vi   Makefile
ifneq ($(KERNELRELEASE),)
    obj-m := ledtest.o
else
    KDIR:=../linux-2.6.39
    PWD:=$(shell pwd)

all:
    $(MAKE) -C $(KDIR) SUBDIRS=$(PWD) modules
endif

clean:
    rm -f *.o *.ko *.mod.c
```

KDIR value indicates the location of the Linux kernel source code. In this example, it is “…/ Linux - 2.6.39”. Please assign to it according to the actual location of the Linux kernel source code.

(2) Compile the driver:

```
# make    ARCH=arm \
CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
```

The compiling process will generate driver file “ledtest.ko” in current directory. In the next section, we will illustrate how to use this driver file.

4.8.4 Download Driver

Copy the “ledtest.ko” file to the corresponding directory for loading. The specific operations are as follows:

(1) Cancel the trigger by other drivers.

```
root@at91sam9x5ek:/# cd    /sys/class/leds/d1
root@at91sam9x5ek:/sys/class/leds/d1# echo    none    >    trigger
root@at91sam9x5ek:/sys/class/leds/d1# cd    ../d2
root@at91sam9x5ek:/sys/class/leds/d2# echo    none    >    trigger
```

(2) Load driver module into kernel

```
root@at91sam9x5ek:/# cd    /media/sda4/MYD-SAM9X5-V2
root@at91sam9x5ek:/media/sda4/MYD-SAM9X5-V2# ls
ledtest.ko      ledtest_app
root@at91sam9x5ek:/media/sda4/MYD-SAM9X5-V2# insmod    ledtest.ko
MYD_SAM9X5_LEDTEST DRIVER MODULE INIT
register MYD_SAM9X5_ledtest Driver OK! Major = 249
MYD-SAM9X5-V2-ledtest initialized
```

At this point, LED driver has been loaded into the kernel successfully. In the next Chapter, we will write a simple application and test the functionality.

4.9 Application Development Example

This Chapter describes the top layer Linux system application development, and a simple instance tells the application development process and driver invocation. The instance achieves the following function: Running the application, the on/off status of two LEDs can be control by passing different parameters.

4.9.1 Source Code Compilation

Create a new directory and a new “ledtest_app.c” file by vi editor in the directory or copy pre-prepared file to current directory directly, The following are the source code of

“ledtest_app.c”.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/ioctl.h>

#define LED_DEV "/dev/MYS-SAM9X5-ledtest"

int main(int argc, char **argv)
{
	int fd, ret, led_num, led_status;
	if (argc!=3 || sscanf(argv[1],"%d", &led_num)!=1
		|| sscanf(argv[2],"%d", &led_status)!=1)
	{
		printf("\r\nPlease input correct parameters !\r\n\n");
		printf("usage:\r\n%s <led_num> <led_status>\r\n", argv[0]);
		printf("\r\nOptions:\r\n");
		printf("    led_num\t-    1 for red led, 0 for blue led.\r\n    led_status\t-    1 for ON, 0
for OFF.\r\n\n");
		exit(1);
	}
	if((led_status!=1 && led_status!=0) || (led_num!=0 && led_num!=1))
	{
		printf("\r\nError: The parameter value must be '0' or '1' !\r\n");
		printf("\r\nPlease try again !!! !\r\n\n");
		exit(1);
	}
	fd = open(LED_DEV, 0);
	if (fd < 0)
	{
		printf("\r\nFail to open device '%s'!\r\n\n", LED_DEV);
		exit(1);
	}

	ret = ioctl(fd, led_status, led_num);
	if(ret < 0)
	{
		printf("\r\nFail calling ioctl !\r\n\n");
	}

	close(fd);

	return 0;
}
```

4.9.2 Compile

There is only one source file in this example for compiling. It is recommended to use “makefile” to manage a project with more than one source file.

Add the path of cross-compiler tools to PATH:

```
# export PATH=$PATH:/usr/local/arm-2010q1/bin/
```

Use the cross compiler tool to compile:

```
# arm-none-linux-gnueabi-gcc  -o  ledtest_app  ledtest_app.c
```

After the above operation, if no error is generated in the current directory, an executable file named “ledtest_app” will be generated.

4.9.3 Application Use

After compilation is completed, it will generate executable file “ledtest_app”, copy to development board by SD card or U disk, and then run file in terminal. Need to pass two parameters when running applications, parameters is used to control two bright LED, the first parameter controls LED (“0" is ON, the "1” is OFF). The second parameter controls which LED lights ("0" is blue led, "1" is red led), the specific operation is as follows:

```
root@at91sam9x5ek:~# /media/sda4/ledtest_app 1  0
MYD-SAM9X5-V2-ledtest Driver Open Called!
MYD_SAM9X5_LED Driver Release Calle
```

After the above operation, blue LED is off.

4.10 Qt Guide

This section describes the development methods and procedures for GUI application useing Qt in MYD-SAM9X5-V2, including two parts. The first part describes how to use the Qt cross compiler tool chain provided by the CD-ROM, general Qt application development only need to use the CD-ROM supplied Qt cross tool chain. The second part describes how the Qt cross tool chain and associated library files be generated by compiling Qt-Embedded source code. Making one’s own Qt development environment is only needed when the Qt library provided when the disc cannot meet the needs of Qt

development.

Note: This image downloaded from “02-Images/Linux/4.3 LCD/Qt” is taken as an example. Please refer to 4.3.4 for automatically download procedures and 4.3.3 for the manually download. The PC environment is Ubuntu 10.04.

4.10.1 Qt Cross Tool Chain

General Qt development only need to use the CD-ROM supplied Qt cross tool chain. The image of the CD-ROM in “02-Images/Linux/4.3 LCD/Qt” has contained the Qt library corresponding to the tool chain, thus the Qt program compiled from the tool chain can directly run on board. Detailed configuration of the Qt cross tool chain provided by the CD-ROM is as follow:

Configuration	Value
Build	Libs
Debug	No
Qt 3 Compatibility	Yes
QtDBus Module	No
QtScriptTools Module	Yes
QtXmlPatterns Module	No
Phonon Module	No
SVG Module	Yes
WebKit Module	Yes
STL Support	Yes
PCH Support	Yes
MMX/3DNOW/SSE/SSE2	No/No/No/No
IWMMXt Support	No
IPv6 Support	Yes
IPv6 Ifname Support	Yes
Getaddrinfo Support	Yes

Configuration	Value
Getifaddrs Support	Yes
Accessibility	Yes
NIS Support	Yes
CUPS Support	No
Iconv Support	No
Glib Support	No
GStreamer Support	No
Large File Support	Yes
GIF Support	Plugin
TIFF Support	Plugin (qt)
JPEG Support	Plugin (qt)
PNG Support	Yes (qt)
MNG Support	Plugin (qt)
Zlib Support	Yes
Session mManagement	No
Embedded Support	Arm
Freetype2 Support	Yes
Graphics (qt)	Linuxfb Multiscreen Linuxfb
Graphics (plugin)	
Decorations (qt)	Styled Windows Default
Decorations (plugin)	
Key board Driver (qt)	Tty Usb
Key board Driver (plugin)	
Mouse Driver (qt)	Pc Linuxtp Pc Linuxtp Tslib
Mouse Driver (plugin)	
OpenGL Support	No
SQLite Support	Qt (qt)
OpenSSL Support	No

Table 4-2

(1) Install Qt cross compile tool to the system directory “/usr/local/”.

Create Qt working directory qt-arm, copy Qt cross compile tool to this directory and decompression

```
$ cd ~
$ mkdir qt-arm
$ cd qt-arm
$ cp /media/cdrom/05-Linux_Source/Qt_Arm/Qt-4.5.3_Tslib-1.4.tar.gz ./
$ sudo tar xvzf Qt-4.5.3_Tslib-1.4.tar.gz -C /usr/local/
```

After decompression, two new directories Qt and tslib will appear in /usr/ local. The Qt directory contains cross compile tool, library and header files. The tslib directory contains touch screen test procedures, libraries and configuration files

(2) Configure system environment

① If you have not add the path of arm-none-linux-gnueabi- to PATH, then you should do it first. The path of arm-none-linux-gnueabi- used in this article is /usr/local/arm-2010q1 /bin, you can use the follow command to add path:

```
$ export PATH=$PATH:/usr/local/arm-2010q1/bin
```

② Set Qt application development related environment variables

The decompressed file “setenv” contains the setting of environment variables, you can use the follow commands to compelet setting:

```
$ source /usr/local/Qt/setenv
```

Or you can enter the settings manually:

```
$ export PATH=$PATH:/usr/localQt/bin
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Qt/lib
```

(3) Burn the image with Qt library

Burn the image in the directory of “02-Images/Linux/4.3 LCD/Qt” in the CD-ROM referring to 4.3.3 for the automatically download procedure and 4.3.4 for the manually download. This image has already established Qt environment and contained all the needed libraries and function modules which are showed in Table 4-2. If the function provided by image can’t match the actual needs, please refer to next section “4.10.2 establish Qt development from source code” to configure our needed function module.

(4) The compiling and running of Qt application program

There are several sample programs under the CD-ROM directory “05-Linux_Source/

Qt_Arm/Qt_Examples/". Here we use masterdetail as an example to tell how to use Qt cross compile tool chain to compile Qt application program and run in the target board.

① Copy the sample program to Qt working directory qt-arm and decompress it

```
$ cd  ~/qt-arm
$ cp  /media/cdrom/05-Linux_Source/Qt_Arm/Qt_Examples/masterdetail.tar.gz  ./
$ tar  xvzf  masterdetail.tar.gz
$ cd  masterdetail
```

② Compile the Qt program (Before compiling, you should ensure that the Qt cross compile chain has been installed and you have already set the environment variables)

```
$ qmake
$ make
```

③ After executing the above command, copy the generated executable file "masterd etail" to development board to run:

```
# ls
masterdetail
# chmod  0777  masterdetail
# ./masterdetail  -qws
```

The result is shown below:

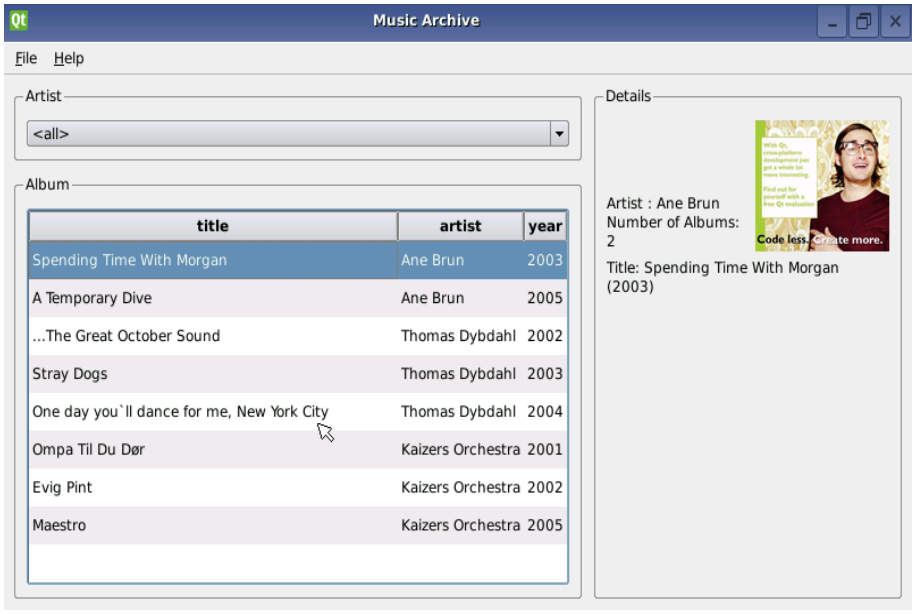


Figure 4-17

4.10.2 Establish Qt Development Environment

This section tell how the Qt cross tool chain and associated library files be generated

by compiling Qt-Embedded source code.

Qt's source code and Tslib's source code are in the CD-ROM under the directory "05-Linux_Source/Qt_Arm/Qt_Source".

(1) Establish a working directory

```
$ cd ~
$ mkdir qt-arm
$ cd qt-arm
```

(2) Compile and install tslib

① Decompress:

```
$ cp /media/cdrom/05-Linux_Source/tslib.tar.gz ./
$ tar xvzf tslib.tar.gz
$ cd tslib
```

② Compile and install:

If you have not added the path of arm-none-linux-gnueabi- to PATH, then you should do it first. The path of arm-none-linux-gnueabi- used in this article is /usr/local/arm-2010q1 /bin, you can use the follow command to add path:

```
$ export PATH=$PATH:/usr/local/arm-2010q1/bin
```

Install the two tools automake and libtool firstly:

```
$ sudo apt-get install automake libtool
```

Configure tslib, you can set up the installation path yourself, here install it into / usr / local / tslib:

```
$ ./autogen.sh
$ ./configure CC=arm-none-linux-gnueabi-gcc CXX=arm-none-linux-gnueabi-g++ --prefix=/usr/local/tslib --the host=arm-linux ac_cv_func_malloc_0_nonnull=yes
```

Compile and install:

```
$ make
$ sudo make install
```

(3) Compile and install qt-embedded :

① Decompress:

In the working directory qt-arm, execute the following commands:

```
$ cp /media/cdrom/05-Linux_Source/qt-embedded-linux-opensource-src-4.5.3.tar.gz\
./
$ tar xvzf qt-embedded-linux-opensource-src-4.5.3.tar.gz
$ cd qt-embedded-linux-opensource-src-4.5.3
```

② Specifies the cross compiler:

Open mkspecs/qws/linux-arm-g++/qmake.conf：

```
$ vi  mkspecs/qws/linux-arm-g++/qmake.conf
```

Open qmake.conf with vi, enter the following commands to replace all the arm-linux- with arm-none-linux-gnueabi-, then save and exit

```
%s/arm-linux-/arm-none-linux-gnueabi-/g
```

③ Configure Qt:

```
$ ./configure -prefix /usr/local/Qt -xplatform qws/linux-arm-g++ -release -opensource -qt-zlib -qt-libtiff -qt-libpng -qt-libmng -qt-libjpeg -make libs -nomake docs -embedded arm -little-endian -qt-freetype -depths 8,16,24  -qt-gfx-linuxfb -qt-kbd-usb -qt-mouse-pc -qt-mouse-linuxtp -qt-mouse-tslib -qt-sql-sqlite -qt3support -I/usr/local/tslib/include -L/usr/local/tslib/lib -confirm-license
```

Enter “/configure - help” to look at parameters description, set parameters by requirement.

④ Compile and install:

```
$ make
$ sudo make install
```

⑤ Set the environment variables:

Run the following command in the current terminal:

```
$ export  PATH=$PATH:/usr/local/Qt/bin
$ export  LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Qt/lib
```

Or add the above commands to /etcprofile file, so that the system will automatically set these environment variables when logging.

(4) Transplant Qt to the development board

① Copy the library to development board

After installing Qt, in order to solve the problem of symbolic links, you can first package, and then directly extracted it to the development board:

```
$ cd  /usr/local/Qt/
$ tar  -zcf  lib.tar.gz  lib
```

Copy the packaged compressed file lib.tar.gz to the development board, and then extract it to /usr/local/Qt:

```
# mkdir  -p  /usr/local/Qt
# tar  xzvf  lib.tar.gz  -C  /usr/local/Qt
```

② Set environment variables

The setting of development board environment variables has been written to “/etc/setqtenv “, it can be completed by the following command:

```
# sourch    /etc/setqtenv
```

Or manually enter settings:

```
# export    QT_QWS_FONTDIR=/usr/local/Qt/lib/fonts
# export    QWS_MOUSE_PROTO=tslib:/dev/input/ touchscreen0
# export    LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/Qt/lib
```

(5) Compiling and running of Qt application program

Please refer to 4.10.1-(4) The compiling and running of Qt application program

Chapter 5 Android System Guide

5.1 Overview

Android is a Linux system based open source operating system, mainly used in portable devices. Android operating system originally developed by Andy Rubin development, initially mainly support mobile phone. In 2005 Android is acquainted by Google, formatting the open mobile phone alliance to improvement it, gradually extended to the Tablet computer and other area. Since its first release Welcomed by the majority of consumers, Android's market shares around the world more than Symbian system for the first time in the first quarter of 2011, ranking first in the world. The data shows that in February 2012, Android accounted for 52.5% of the share of the global smartphone operating system market.

Android system is running based on Linux system, mainly made by Linux Kernel, system libraries, Dalvik virtual machine, application framework, and applications written mainly by JAVA. Its framework is as shown in Figure 5-1:

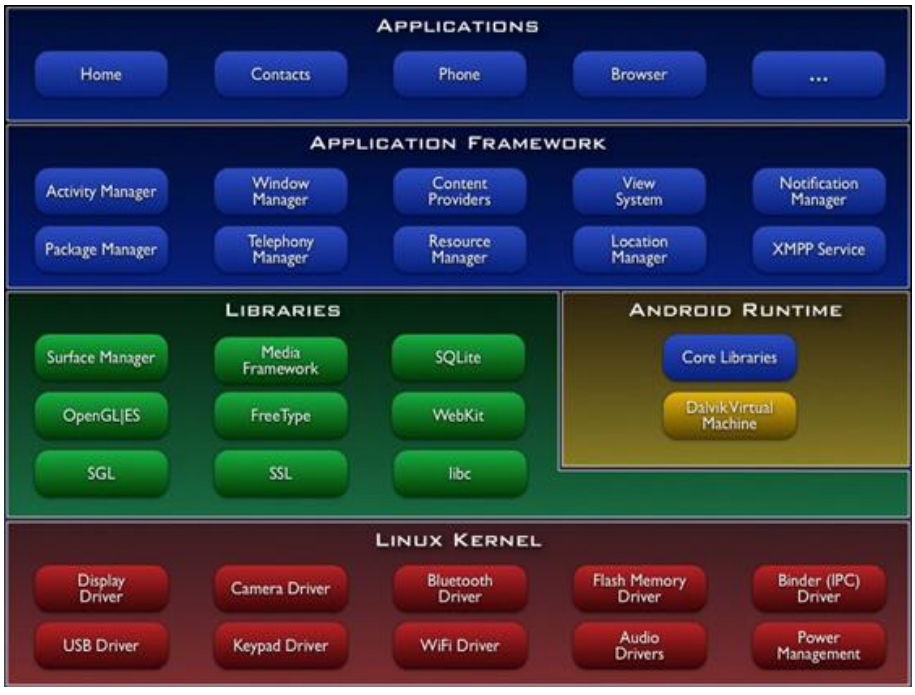


Figure 5-1

This Chapter describes how to build and run Android 2.3.5 system in MYD-SAM9X5-

V2 platform, include the following main content:

(1) Build Android System

(2) Compile Android

(3) Android System Use

5.2 Software Resources

Software resources are shown in Table 5-1:

Category	Name	Note
Boot program	AT91Bootstrap	Use to guide U-boot
	U-boot	1.Support NandFlash Erase, Read and Write 2. Support Network to Download Image 3. Support Settings, Save Environment Variable 4. Support Display, Contrast, Modify Memory Content 5. Support Bootm, Bootargs Settings
Linux Kernel	Linux 2.6.39	Develop Linux kernel for MYD-SAM9X5-V2 hardware
Device Drivers	Network Port Driver	ETH0 Provide Source
	Serial Port Driver	USART0, DBGU, Provide Source
	USB	USB_THE HOST*2, USB_OTG, Provide Source
	SMD Driver	Only Provide Hardware Port
	SD Card Driver	Micro-SD, SD-Card, Provide Source
	LCD+ Touch	LCD Driver, Provide Source
	SPI Driver	Provide Source
	TWI Driver	Two Wire Port, I2C, Provide Source
	DMA Driver	Have Been Tested to Provide Source
	GPIO Driver	Have Been Tested to Provide Source
System Files	Android System Files	Have Been Tested to Provide Binary Image File

Table 5-1

5.3 Build Android System

This Chapter describes how to use image to build Android system.

5.3.1 Install Download Tool

(1) Install Atmel ISP download software SAM-BA (2.11 or later, CD-ROM location: 03-Tools/SAM-BA/) (Note: If install SAM-BA 2.10 and earlier versions, Please uninstall previous versions and USB driver). If need two or more SAM-BA version coexistence, different SAM-BA versions use different USB port.

(2) Power on and connect the board (J8) to PC by micro USB cable.

(3) Turn SW1_1, SW1_2 off, disconnect Base board jumper JP8, and then press NRST button to reset board (the order is not reversed). Firstly, connect PC to install driver, then select the location to install SAM-BA installation directory, Refer to Figure 5-2:

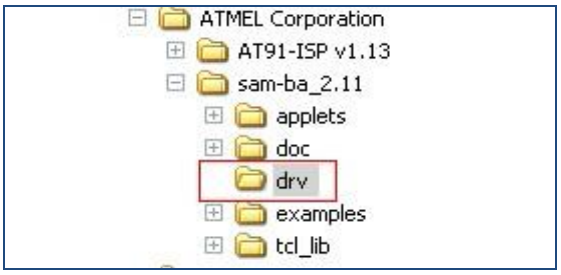


Figure 5-2

(4) If there is prompt in device manager in Figure 5-3, it shows board driver has been installed.

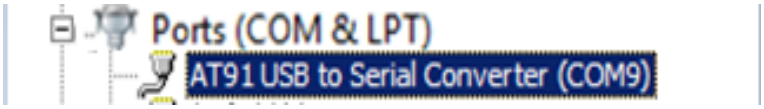


Figure 5-3

The COM9 is connection port (determined by actual situation, here for COM9).

5.3.2 Connect the Board To SAM-BA

(1) Install MYD-SAM9X5-V2 USB driver

Please refer to “03-Tools\SAM-BA\the” board driver install.pdf.

(2) Connect board. The specific steps are as follows:

① Connect the board to PC by USB line

② Turn SW1_1, SW1_2 off.

③ Press NRST to reset board, turn SW1_1 on.

At this point, there will be USB equipment.

(3) Connect development board USB_UART and PC using USB cable, and set up super terminal parameters for: COM1, 115200, 8, none, 1. The COM port setting according to actual situation. The detailed steps for reference: 3.2.3 super terminal configuration and ready to download

5.3.3 Automatic Download

Note: 1. Pull out SD card, before download.

2. Here to use the 4.3-inch screen image as an example, if using a different screen sizes, please download the corresponding directory image.

Complete Chapter 5.3.1 and 5.3.2, open the disk directory “02-Images/Android/4.3 LCD/” and edit “at91sam9x5.bat” file. Refer to Figure 5-4:

Figure 5-4

The original C0Mx changes correspondence connection port as COM9. Double-click “at91sam9x5.bat”, Android image begins to download board automatically, waiting 2-3 minutes, it will pop up “logfile.log” file automatic which represents automatic writing is completed.

5.3.4 Manual Download

Note: 1. Pull out SD card, before download

2. Here to use the 4.3-inch screen image as an example, if using a different screen sizes, please download the corresponding directory image.

All image files used in this Chapter can be found in the directory: “02-Images/Android/4.3 LCD/”.

The NandFlash content of Android system is divided as shown in Figure 5-5:

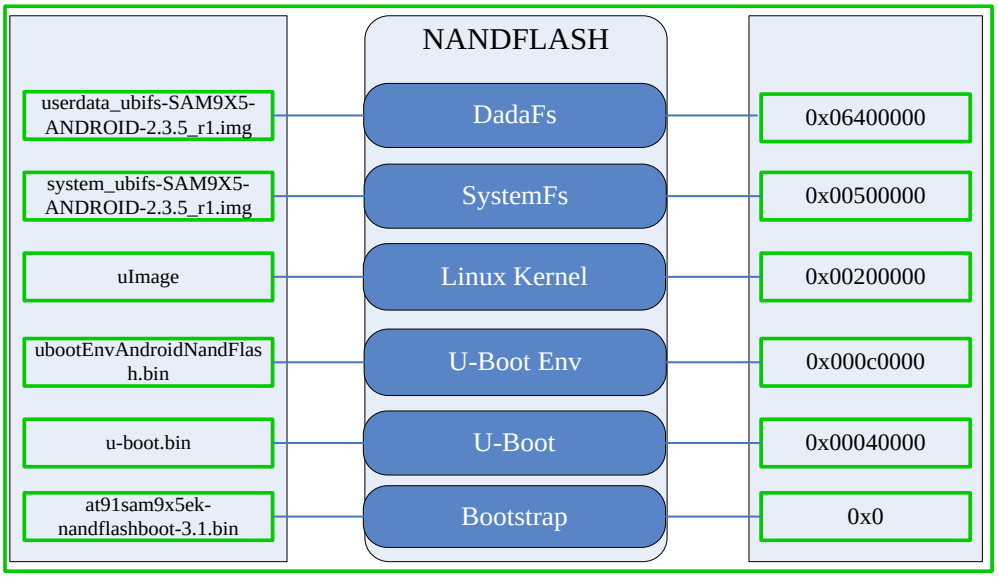


Figure 5-5

Download Linux by SAM-BA manually

(1) Complete Chapter 5.3.1 and 5.3.2, double-click SAM-BA v2.11, there appears port. Refer to Figure 5-6:

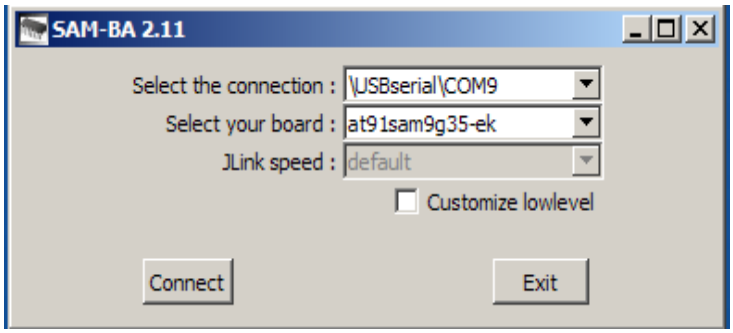


Figure 5-6

Click “Connect” to enter SAM-BA port. Refer to Figure 5-7:

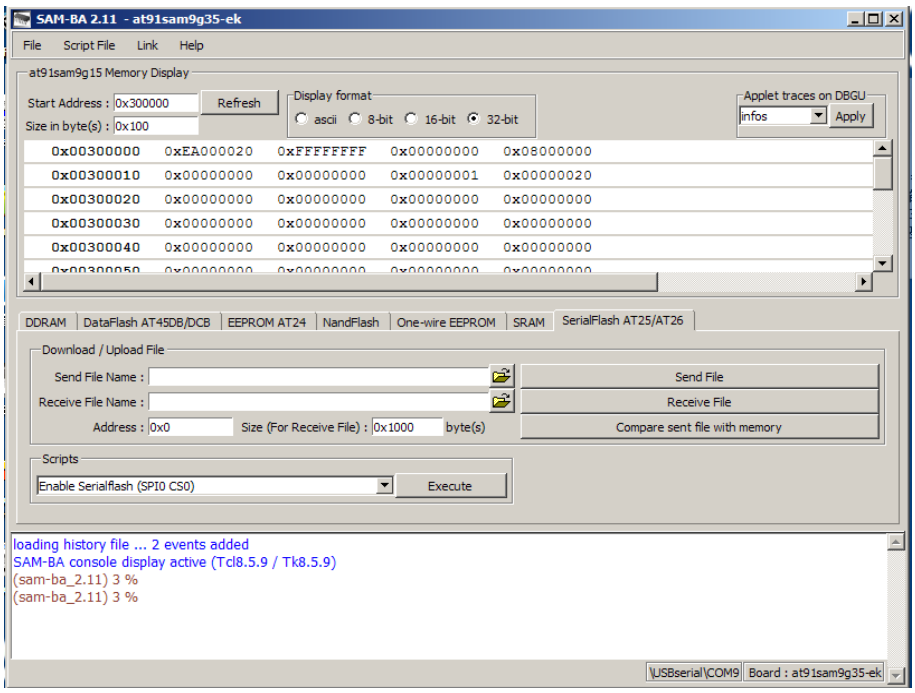


Figure 5-7

(2) Select “NandFlash” tab, Enable NandFlash in Scripts tab and then click “Execute”.

Refer to Figure 5-8:

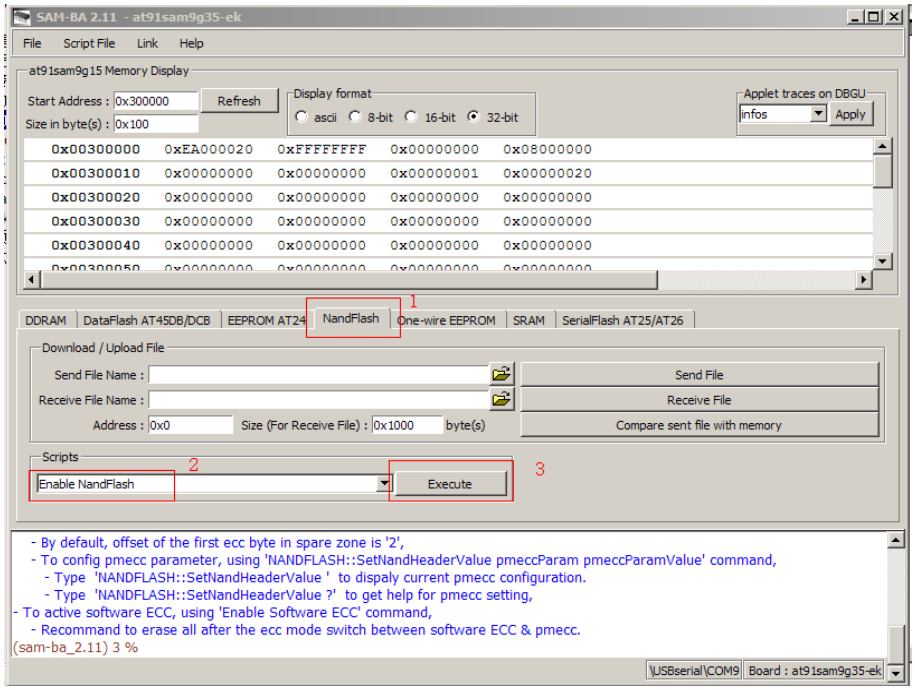


Figure 5-8

(3) Select Enable “OS PMECC” parameters in Scripts tab, then click Execute, using the default option, click “OK” directly (Note: there cannot check “Trimffs”). Refer to Figure 5-9:

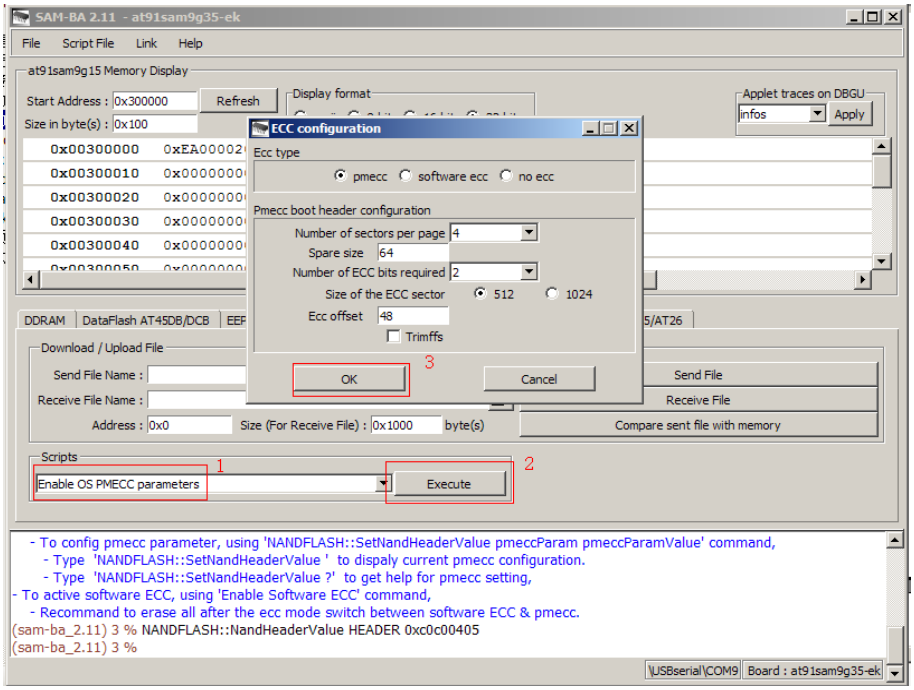


Figure 5-9

(4) Select Erase All in Scripts tab and then click Execute. Refer to Figure 5-10:

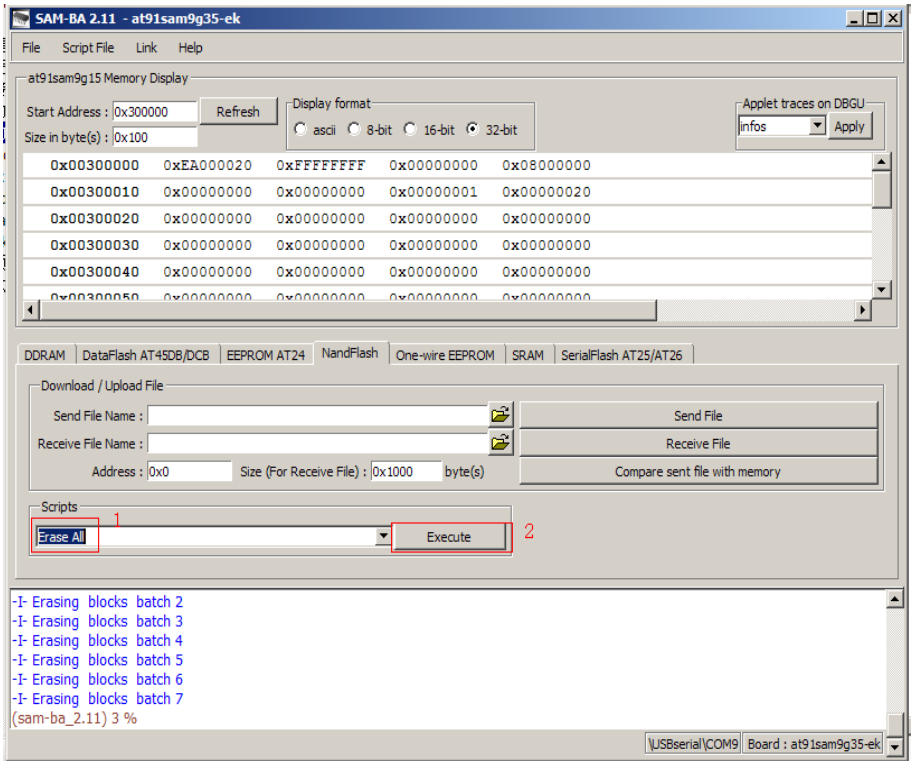


Figure 5-10

(5) Download “at91sam9x5ek-nandflashboot-3.1.bin”. Refer to Figure 5-11, 5-12:

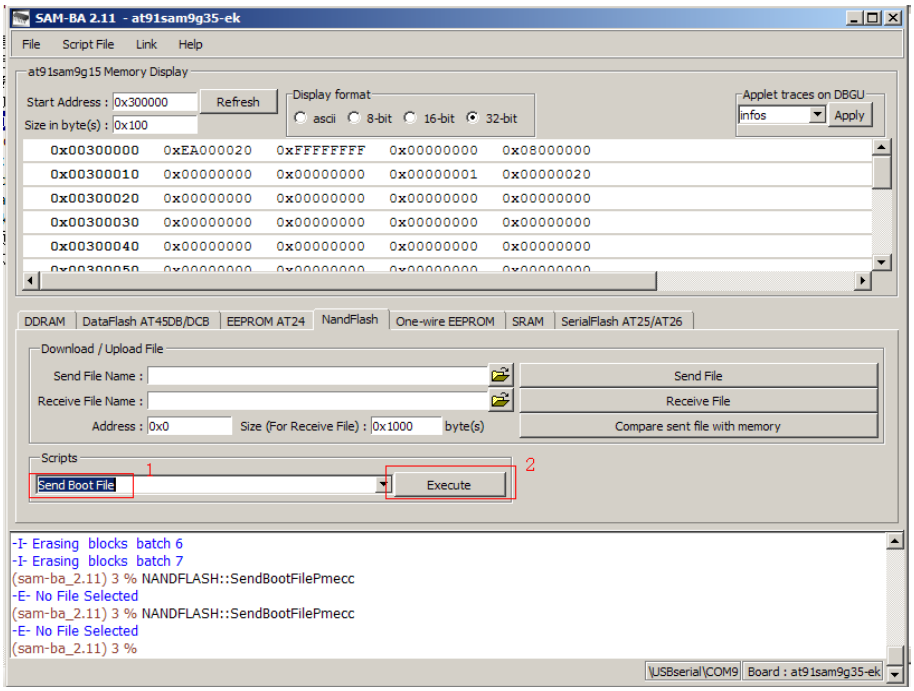


Figure 5-11

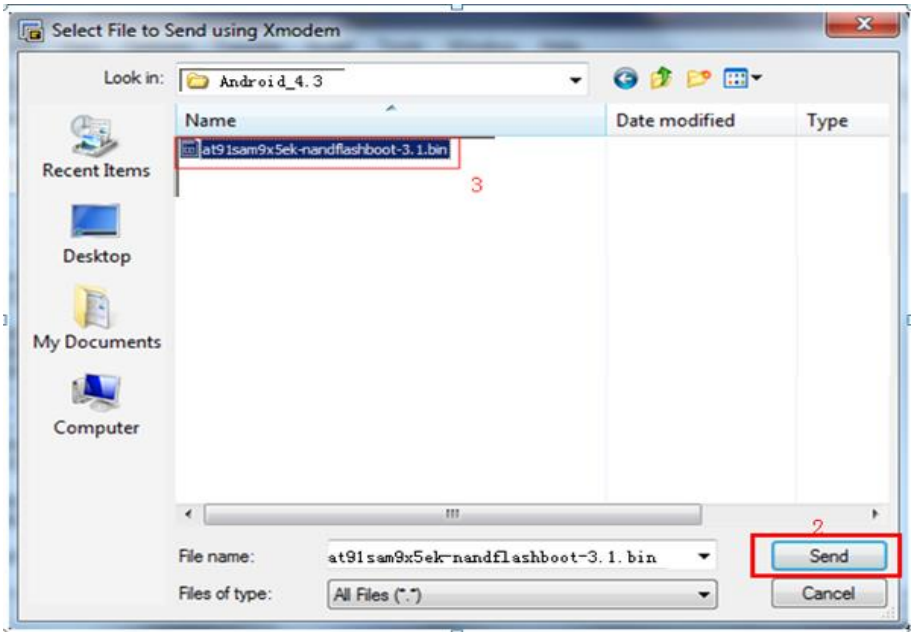


Figure 5-12

(6) Download “u-boot.bin” file to 0x40000 Department. Refer to Figure 5-13:

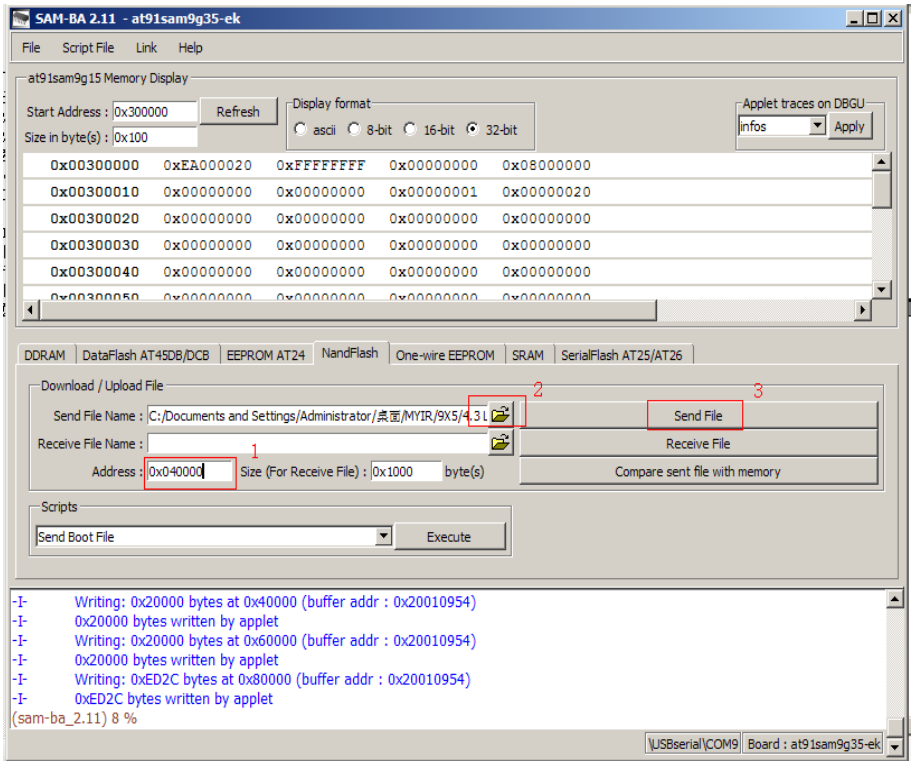

SAM-BA 2.11 - at91sam9g35-ek
File Script File Link Help
at91sam9g15 Memory Display
Start Address : 0x300000
Size in byte(s) : 0x100
Refresh
Display format
ascii 8-bit 16-bit 32-bit
Applet traces on DBGU
infos
Apply
0x00300000 0xEA000020 0xFFFFFFFF 0x00000000 0x08000000
0x00300010 0x00000000 0x00000000 0x00000001 0x00000020
0x00300020 0x00000000 0x00000000 0x00000000 0x00000000
0x00300030 0x00000000 0x00000000 0x00000000 0x00000000
0x00300040 0x00000000 0x00000000 0x00000000 0x00000000
DDRAM DataFlash AT45DB/DCB EEPROM AT24 NandFlash One-wire EEPROM SRAM SerialFlash AT25/AT26
Download / Upload File
Send File Name : C:/Documents and Settings/Administrator/桌面/MYIR/9X5/4.3 L
Receive File Name :
Address : 0x40000
Size (For Receive File) : 0x1000 byte(s)
Send File
Receive File
Compare sent file with memory
1
2
3
Scripts
Send Boot File
Execute
-I- Writing: 0x20000 bytes at 0x40000 (buffer addr : 0x20010954)
-I- 0x20000 bytes written by applet
-I- Writing: 0x20000 bytes at 0x60000 (buffer addr : 0x20010954)
-I- 0x20000 bytes written by applet
-I- Writing: 0xED2C bytes at 0x80000 (buffer addr : 0x20010954)
-I- 0xED2C bytes written by applet
(sam-ba_2.11) 8 %
\USBserial\COM9
Board : at91sam9g35-ek

Figure 5-13

(7) Download “U-bootEnvAndroidNandFlash.bin” to 0xc0000. Refer to Figure 5-14:

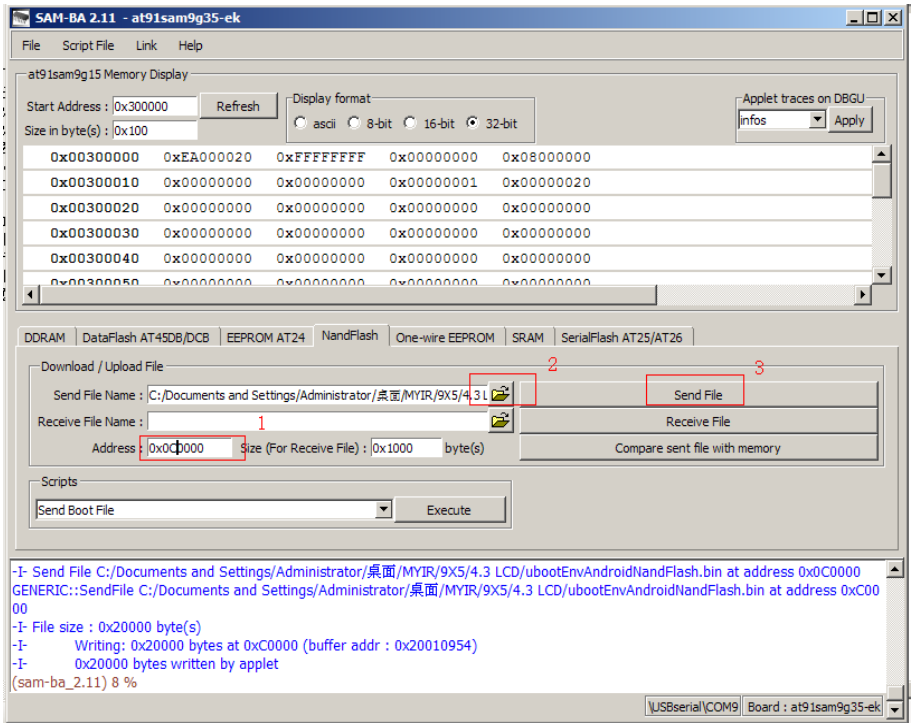

SAM-BA 2.11 - at91sam9g35-ek
File Script File Link Help
at91sam9g15 Memory Display
Start Address : 0x300000
Size in byte(s) : 0x100
Refresh
Display format
ascii 8-bit 16-bit 32-bit
Applet traces on DBGU
infos
Apply
0x00300000 0xEA000020 0xFFFFFFFF 0x00000000 0x08000000
0x00300010 0x00000000 0x00000000 0x00000001 0x00000020
0x00300020 0x00000000 0x00000000 0x00000000 0x00000000
0x00300030 0x00000000 0x00000000 0x00000000 0x00000000
0x00300040 0x00000000 0x00000000 0x00000000 0x00000000
DDRAM DataFlash AT45DB/DCB EEPROM AT24 NandFlash One-wire EEPROM SRAM SerialFlash AT25/AT26
Download / Upload File
Send File Name : C:/Documents and Settings/Administrator/桌面/MYIR/9X5/4.3 L
Receive File Name :
Address : 0x0C0000
Size (For Receive File) : 0x1000 byte(s)
Send File
Receive File
Compare sent file with memory
1
2
3
Scripts
Send Boot File
Execute
-I- Send File C:/Documents and Settings/Administrator/桌面/MYIR/9X5/4.3 LCD/ubootEnvAndroidNandFlash.bin at address 0x0C0000
GENERIC::SendFile C:/Documents and Settings/Administrator/桌面/MYIR/9X5/4.3 LCD/ubootEnvAndroidNandFlash.bin at address 0xC0000
-I- File size : 0x20000 byte(s)
-I- Writing: 0x20000 bytes at 0xC0000 (buffer addr : 0x20010954)
-I- 0x20000 bytes written by applet
(sam-ba_2.11) 8 %
\USBserial\COM9
Board : at91sam9g35-ek

Figure 5-14

(8) Download Linux kernel u-Image to 0x200000. Refer to Figure 5-15:

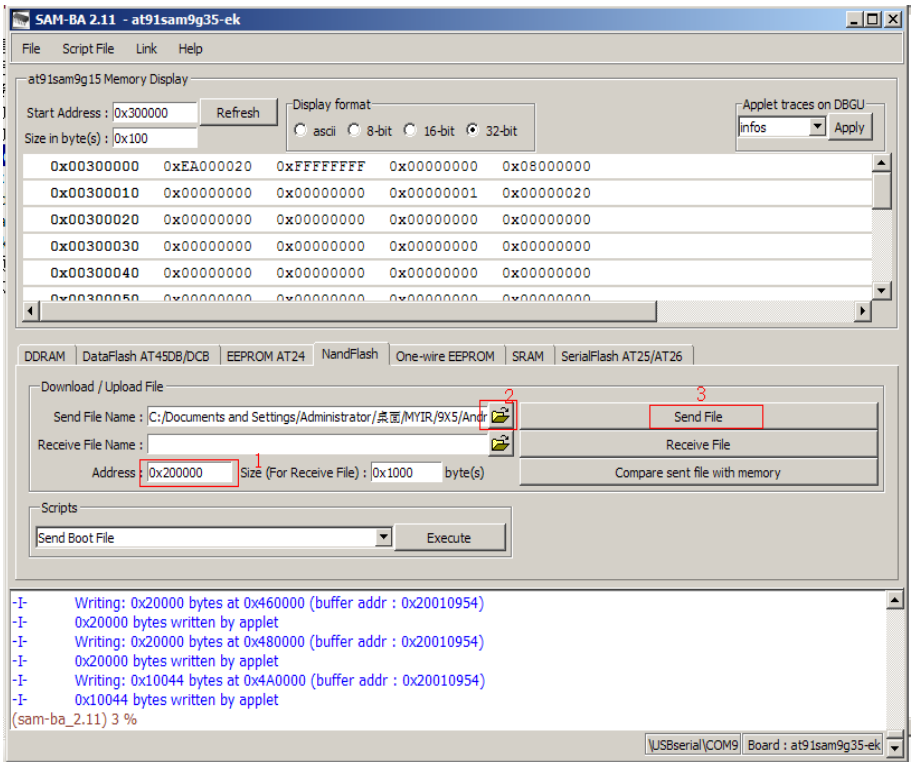

SAM-BA 2.11 - at91sam9g35-ek
File Script File Link Help
at91sam9g15 Memory Display
Start Address : 0x300000
Size in byte(s) : 0x100
Refresh
Display format
ascii 8-bit 16-bit 32-bit
Applet traces on DBGU
infos
Apply
0x00300000 0xEA000020 0xFFFFFFFF 0x00000000 0x08000000
0x00300010 0x00000000 0x00000000 0x00000001 0x00000020
0x00300020 0x00000000 0x00000000 0x00000000 0x00000000
0x00300030 0x00000000 0x00000000 0x00000000 0x00000000
0x00300040 0x00000000 0x00000000 0x00000000 0x00000000
DDRAM DataFlash AT45DB/DCB EEPROM AT24 NandFlash One-wire EEPROM SRAM SerialFlash AT25/AT26
Download / Upload File
Send File Name : C:/Documents and Settings/Administrator/桌面/MYIR/9X5/Andr
Receive File Name :
Address : 0x200000
Size (For Receive File) : 0x1000 byte(s)
Send File
Receive File
Compare sent file with memory
1
2
3
Scripts
Send Boot File
Execute
-I- Writing: 0x20000 bytes at 0x460000 (buffer addr : 0x20010954)
-I- 0x20000 bytes written by applet
-I- Writing: 0x20000 bytes at 0x480000 (buffer addr : 0x20010954)
-I- 0x20000 bytes written by applet
-I- Writing: 0x10044 bytes at 0x4A0000 (buffer addr : 0x20010954)
-I- 0x10044 bytes written by applet
(sam-ba_2.11) 3 %
\USBserial\COM9
Board : at91sam9g35-ek

Figure 5-15

(9) Before download file system, first enable “Trimffs” (Note: Be sure to check “Trimffs”). Refer to Figure 5-16:

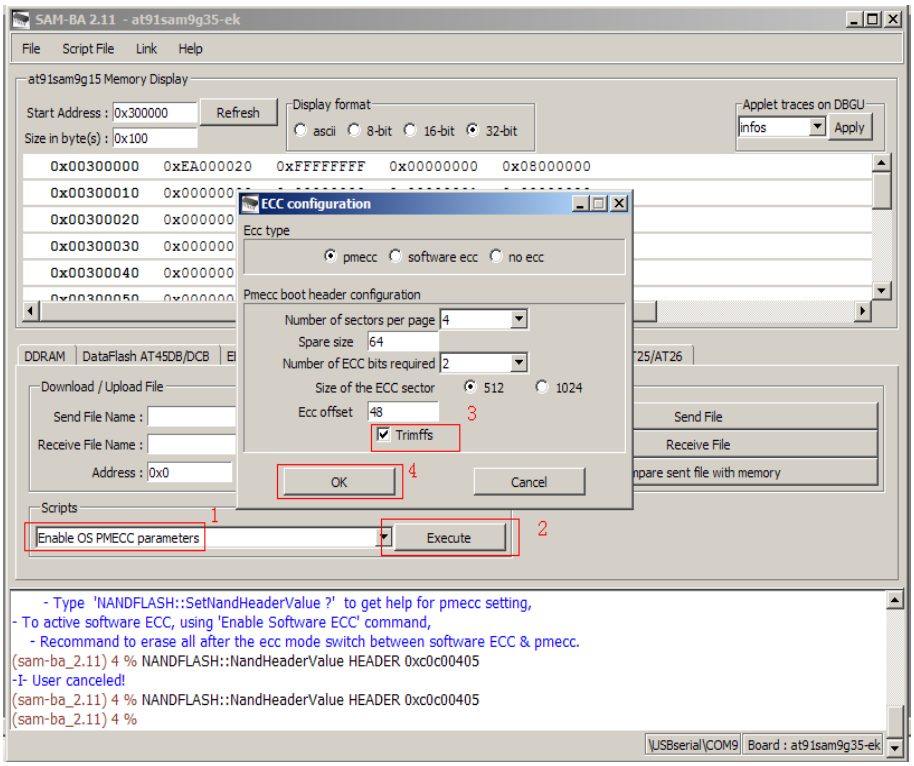


Figure 5-16

(10) Download “system_ubifs-SAM9X5-ANDROID-2.3.5_r1.img” to 0x500000. Refer

to Figure 5-17:

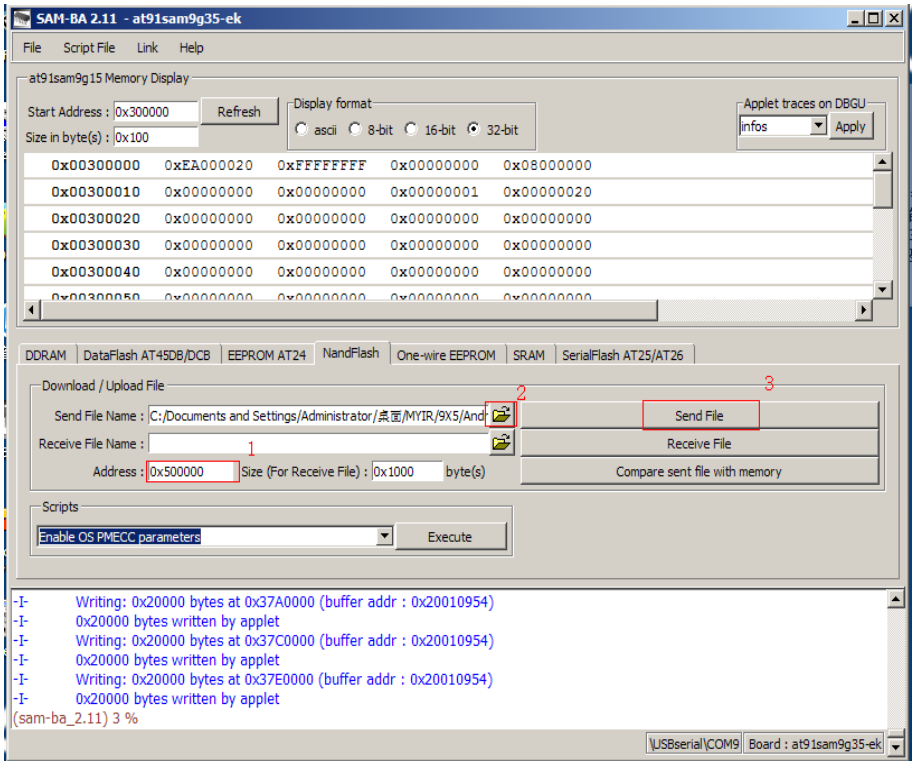


Figure 5-17

(11) Download "userdata_ubifs-SAM9X5-ANDROID-2.3.5_r1.img" to 0x6400000. Refer to Figure 5-18:

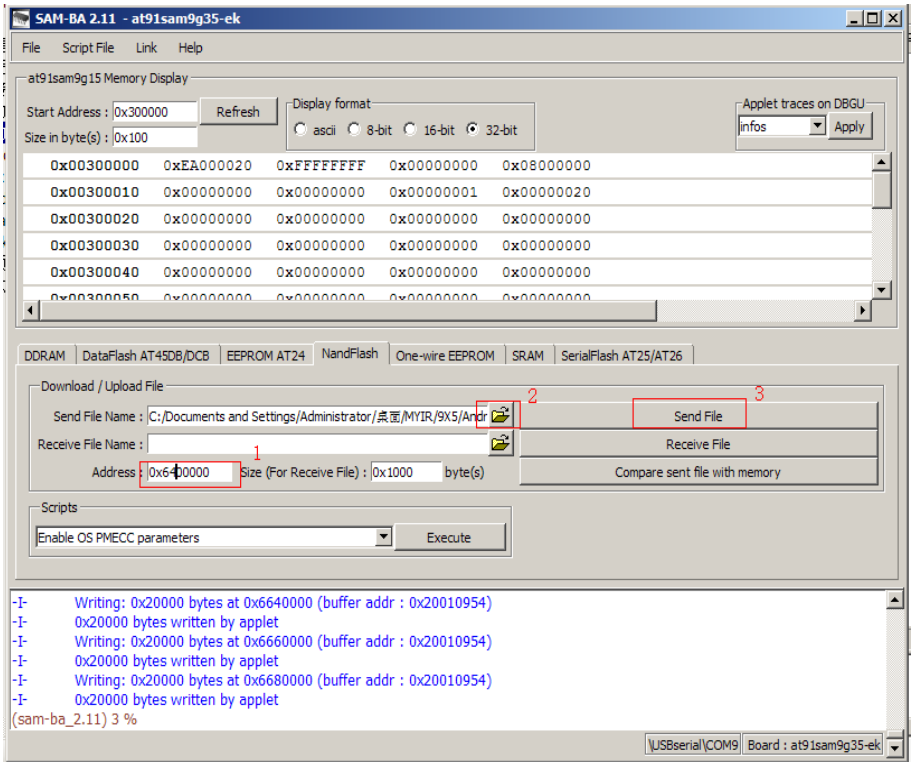


Figure 5-18

At this point, Android system image file download is completed, and press K1 key can restart Android system.

5.4 Compile Android System Files

This Chapter will describe the compiled methods and steps of Android system files.

5.4.1 Android System Principle

(1) File description:

File	Description
“at91sam9x5ek-nandflashboot-3.1.bin”	Boot program Compiled by AT91Bootstrap source is used to start u-boot
“u-boot.bin”	The secondary boot for boot kernel
“u-Image”	Linux kernel file compiled by Linux kernel source code
“system_ubifs-SAM9X5-ANDROID-2.3.5_r1.img”	Android File System (system files Chapter)
“userdata_ubifs-SAM9X5-ANDROID-2.3.5_r1.img”	Android File Systems (user data portion)
“at91sam9x5.tcl”	Writing log file and view it by notebook
“at91sam9x5ek_demo_android_nandflash.bat”	Automatic programming tools (MS - DOS batch file, the manual programming process does not require this file)

Table 5-2

(2) The principle of the system

Power on, when system starts form nandflash, start steps is as following:

① The boot code “ at91sam9x5ek-nandflash boot-3.1.bin” in nandflash will be copied by at91sam9x5 internal rom’s boot code to SRAM and been executed. Bootloader initializes hardware, include setting CPU frequency, configure running configuration PIO, and then copy the secondary boot program “U-boot.bin” to DDRAM for execution.

② Secondary bootloader U-boot is mainly responsible for boot Linux, including set Linux operating environment, Load Linux image file “u-Image”, pass startup parameters to

Linux, last boot Linux to start .

③ Android system will start when linux boot up and file system mounted

5.4.2 Compile System Files

The Android is based on Linux, please setup Linux compiling tools chain platform for developing Android.

(1) Install Cross Compiler Tools

Decompress cross compiler tool to “/usr/local directory”:

```
# tar xvjf   \
05-Linux_Source/CrossTool/ arm-2010q1-202-arm-none-linux-gnueabi.tar.bz2 \
-C /usr/local
```

(2) Compile AT91Bootstrap

```
# tar xjvf 05-Linux_Source/AT91Bootstrap/AT91Bootstrap-5series_1.2.tar.bz2
# cd AT91Bootstrap-5series_1.2
# make at91sam9x5nf_defconfig
# make distclean
# make CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
# cd binaries
```

In this directory, “at91sam9x5ek-nandflashboot-3.1.bin” is AT91Bootstrap.

(3) Compile U-boot

Note: The default compiled U-boot without debug function will boot kernel directly and skipped network configuration.

```
# tar xjvf 05-Linux_Source/U-Boot/u-boot-2010.06.tar.bz2
# cd u-boot-2010.06
# make at91sam9x5ek_nandflash_config
# make CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
```

There will be “u-boot.bin” in “u-boot-2010.06” directory when compilation is completed.

(4) Compile Linux Kernel

Decompress Linux kernel to working directory:

```
# tar xvjf 06-Android_Source/Linux_Kernel_For_Android/linux-2.6.39.tar.bz2
# cd linux-2.6.39/
```

Configure File (Select a different configuration file depending on the LCD size).

LCD Model	Profile
LCD_4.3	MYIR_MYD-SAM9X5-V2_4.3LCD_Android_defconfig
LCD_7.0	MYIR _MYD-SAM9X5-V2_7.0LCD_Android_defconfig
LCD_10.2	MYIR _MYD-SAM9X5-V2_10.2LCD_Android_defconfig

Compile it automatically or manually:

① Executed source code under the root directory to compile the script make_image.sh.

```
# sudo chmod a+x make_image.sh
# ./make_image.sh 4.3
```

Where 4.3 represents using a 4.3 -inch screen, the screen if you use other dimensions, please change corresponding size.

② Manual Compile

Configure default Application. Here in 4.3-inch LCD screen configuration, for example:

```
# make ARCH=arm myir_MYD-SAM9X5_4.3LCD_Android_defconfig
```

Copy boot logo file. "Initlogo_480x272.rle" for the 4.3 -inch screen boot logo file. If it is 7.0 or 10.2 inches, please use "initlogo_800x480.rle".

Execute following command to compile Linux kernel:

```
# make uImage ARCH=arm \
CROSS_COMPILE=/usr/local/arm-2010q1/bin/arm-none-linux-gnueabi-
```

Note: make uImage command requires compile environment installed U-boot-mkimage tool; otherwise, use the following command to install tool:

 # apt-get install U-boot-mkimage

After compile kernel, u-Image is Linux kernel in directory “arch/arm/boot/”.

5.5 Android System Application

5.5.1 USB Key Board Test

Insert USB key board to J15, press the “NumLock” key, when the lights of the key board in the upper-right corner turn green, it shows USB key board can be used.

5.5.2 Browse Picture Test

(1) Select "Gallery" icon, Gallery port will pop up if insert SD card. Refer to Figure 5-19:

Figure 5-19

(2) Select a picture folder. Refer to Figure 5-20:

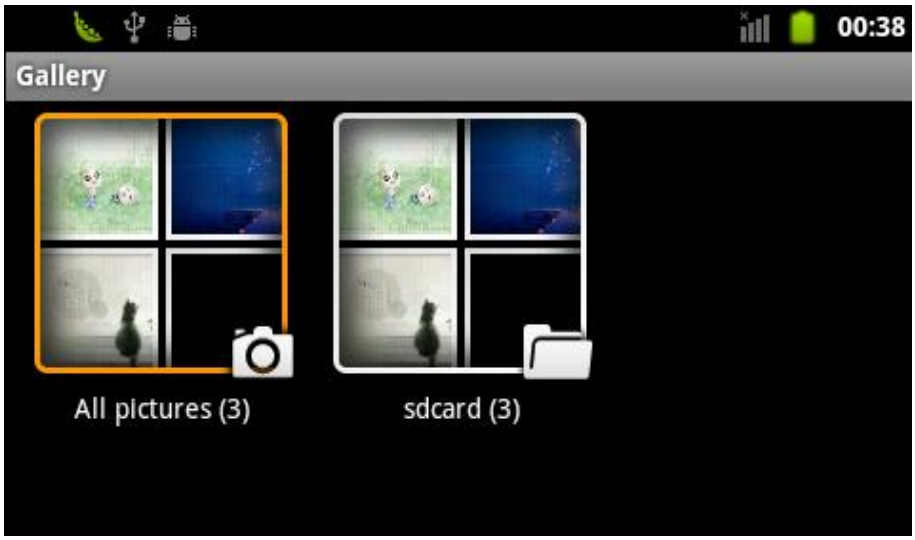


Figure 5-20

(3) View picture, the results are shown in Figure 5-21:

Figure 5-21

5.5.3 Play Audio Test

(1)Select "Music" icon will pop up music player port. Refer to Figure 5-22:

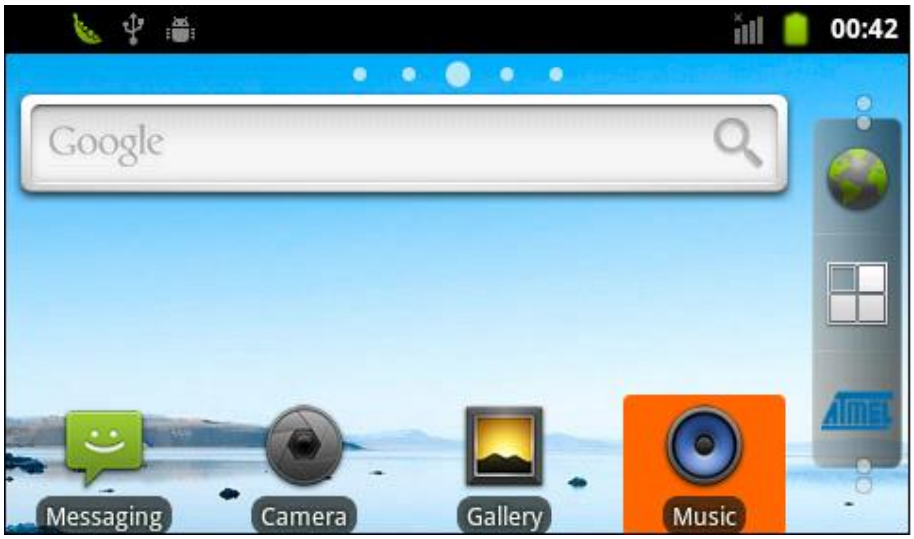


Figure 5-22

(2) Select "Songs" option. Refer to Figure 5-23:

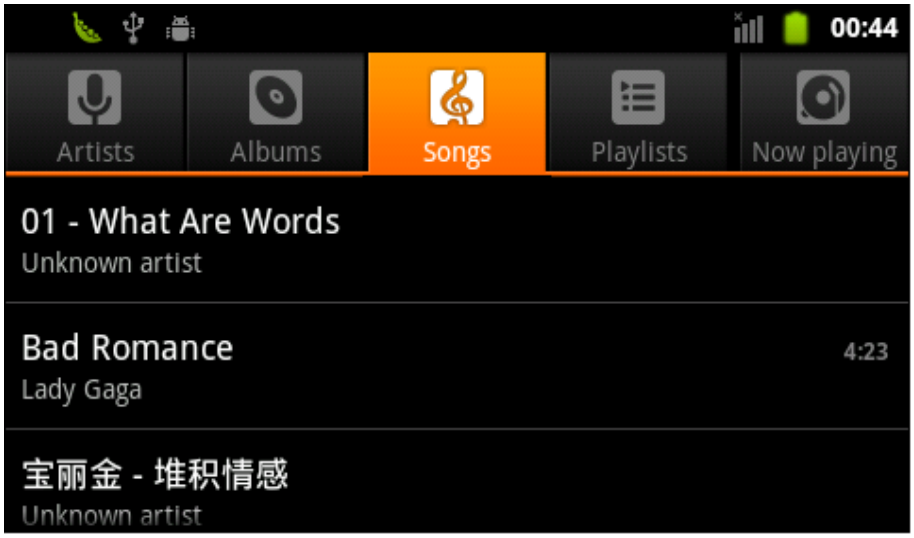


Figure 5-23

(3) Select a song to play. The effect is as shown in Figure 5-24:

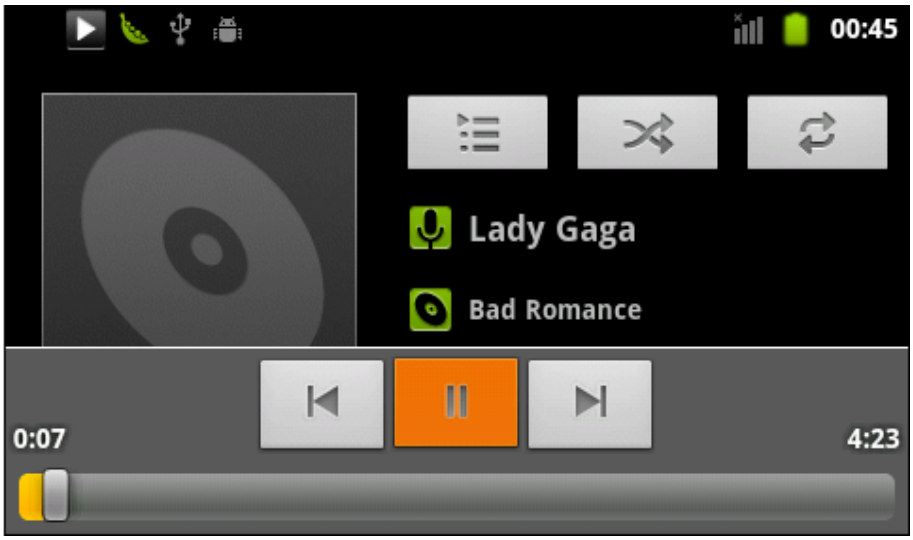


Figure 5-24

5.5.4 Ethernet Test

Note: Firstly, connect the board to router by cable

(1) Enter port and select icon "Ethernet". Refer to Figure 5-25:

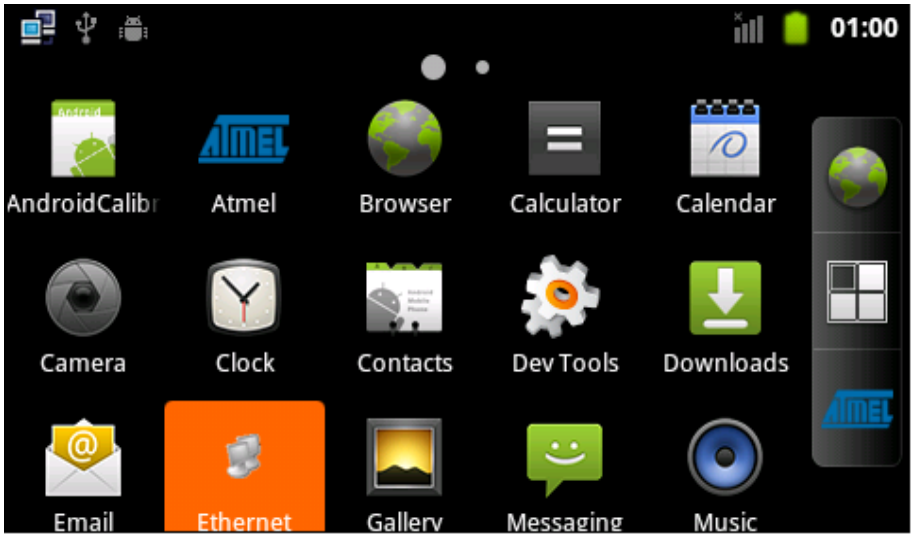


Figure 5-25

(2) Open Ethernet. Refer to Figure 5-26:

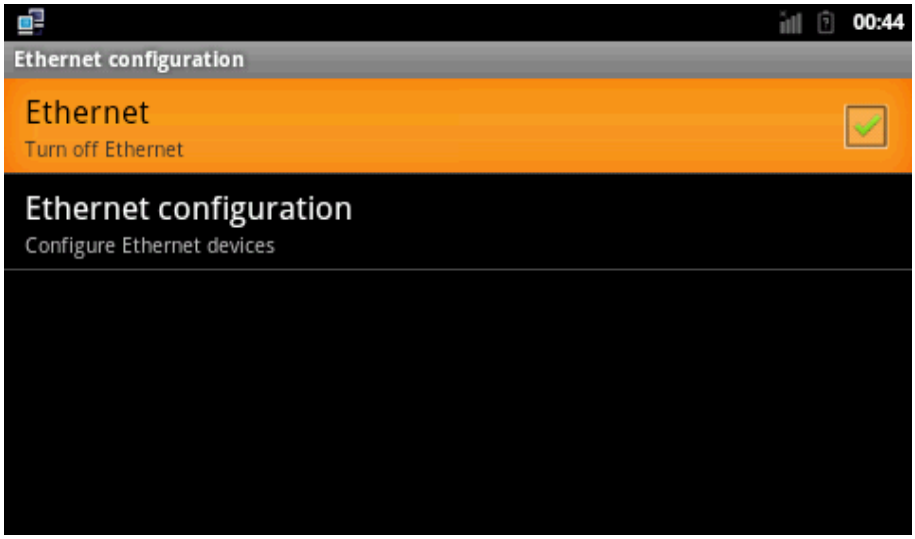


Figure 5-26

(3) Configure Ethernet. Refer to Figure 5-27:

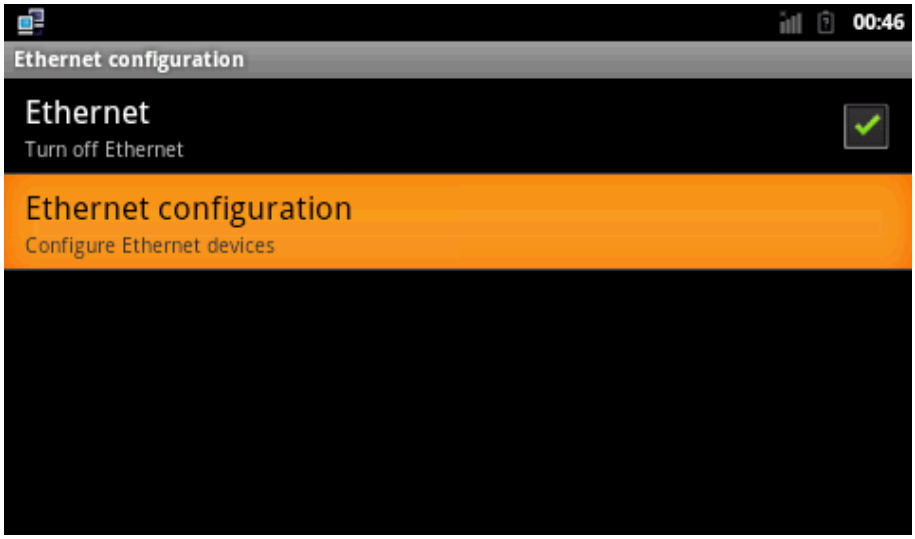


Figure 5-27

(4) Select "Dhcp" to obtain dynamic IP. Otherwise, select "Static IP" manually to set IP address, subnet mask, DNS server, default gateway. Refer to Figure 5-28:

Figure 5-28

（5）After configuration is successful, input string: www.baidu.com. Refer to Figure 5-29:

Figure 5-29

Appendix 1 FAQ

Q1: Report "Connection \USBserial\COMxx not found" (XX: port number, and according to the situation such as the host machine is COM13, then XX is 13), pop up logfile file contents. As shown below:

```
-I- Waiting ...
-I- TCL platform : Windows NT
-I- SAM-BA 2.11    on : windows
-I- Retrieved arguments from command line :
-I- argv 0 : \USBserial\COM13
-I- argv 1 : at91sam9x35-ek
-I- argv 2 : at91sam9x35ekes_test_demo.tcl
-E- Connection \USBserial\COM13 not found
-E- Connection list : COM1
```

Analysis and Answers:

This problem happens because samba connection cannot be found "\ USBserial \ COMxx", DIP switch SW1_1 and SW1_2 off in the control panel, disconnect JP8 jumper (CAN0_RX_EN), press NRST reset board , turn SW1_2 on, and then start download.

Display “logfile” file as follows:

```
-I- Waiting ...
-I- TCL platform : Windows NT
-I- SAM-BA 2.11    on : windows
-I- Retrieved arguments from command line :
-I- argv 0 : \USBserial\COM3
-I- argv 1 : at91sam9x35-ek
-I- argv 2 : at91sam9x35ekes_test_demo.tcl
-E- Connection \USBserial\COM3 not found
-E- Connection list : {\USBserial\COM13} COM1
```

It indicates that port isn’t right, as above -I - argv 0: “\ USBserial \ COM3 " , but connections list is "-E-Connection list: {\ USB serial \ COM13} COM1 " which shows native port is COM13 not COM3 and it needs to modify COM port (Note: COM port is that your the host use).

Q2: Prompt "Can't detect known device"

Analysis and Answers:

This phenomenon isn't turn on the corresponding switch when enable Serialdataflash or Nandflash. Enable Serial dataflash without turning SW1_2on, enable Nandflash without turning SW1_1 on (whether manual download or automatic download, it will have a prompt. Automatic download is recorded in logfile).

Q3: Download system successfully, but system boot failed.

Analysis and Answers:

If downloading linux image to board with SD card mounted, the system might fail to start and show following message on Hyper Terminal. Please unplug SD card and try to download linux image again.

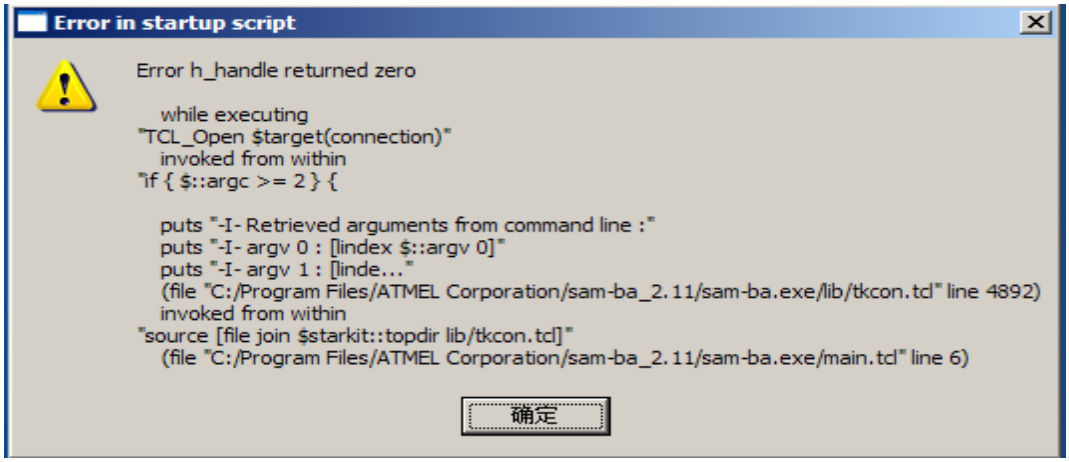
```
NAND read: device 0 offset 0x200000, size 0x250000
atmel_nand : one bit error on data. (data byte : b8, in page offset : 368, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 71, in page offset : 1, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : f9, in page offset : 160, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 60, in page offset : 437, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : 74, in page offset : 436, bit offset : 0x5)
atmel_nand : error corrected
atmel_nand : one bit error on data. (data byte : a0, in page offset : 340, bit offset : 0x5)
atmel_nand : error corrected
 2424832 bytes read: OK
## Booting kernel from Legacy Image at 22000000 ...
Bad Header Checksum
ERROR: can't get kernel image!
U-Boot>
```

Q4: Automatic download for a long time, HyperTerminal did not continue to output download information.

Analysis and Answers:

This may be stuck in automatic download process. It can end sam-ba.exe process in task manager and then restart download.

If start SAM-BA v2.11, Click Connect and pops up the following window when in automatic download:


Error in startup script
Error h_handle returned zero
while executing
"TCL_Open $target(connection)"
invoked from within
"if { $::argc >= 2 } {
puts "-I- Retrieved arguments from command line :"
puts "-I- argv 0 : [lindex $::argv 0]"
puts "-I- argv 1 : [linde..."
(file "C:/Program Files/ATMEL Corporation/sam-ba_2.11/sam-ba.exe/lib/tkcon.tcl" line 4892)
invoked from within
"source [file join $starkit::topdir lib/tkcon.tcl]"
(file "C:/Program Files/ATMEL Corporation/sam-ba_2.11/sam-ba.exe/main.tcl" line 6)
确定

This failure might cause by duplicated running SAM-BA, please end duplicated SAM-BA process and try again.

Q5: MDK routines cannot debug online

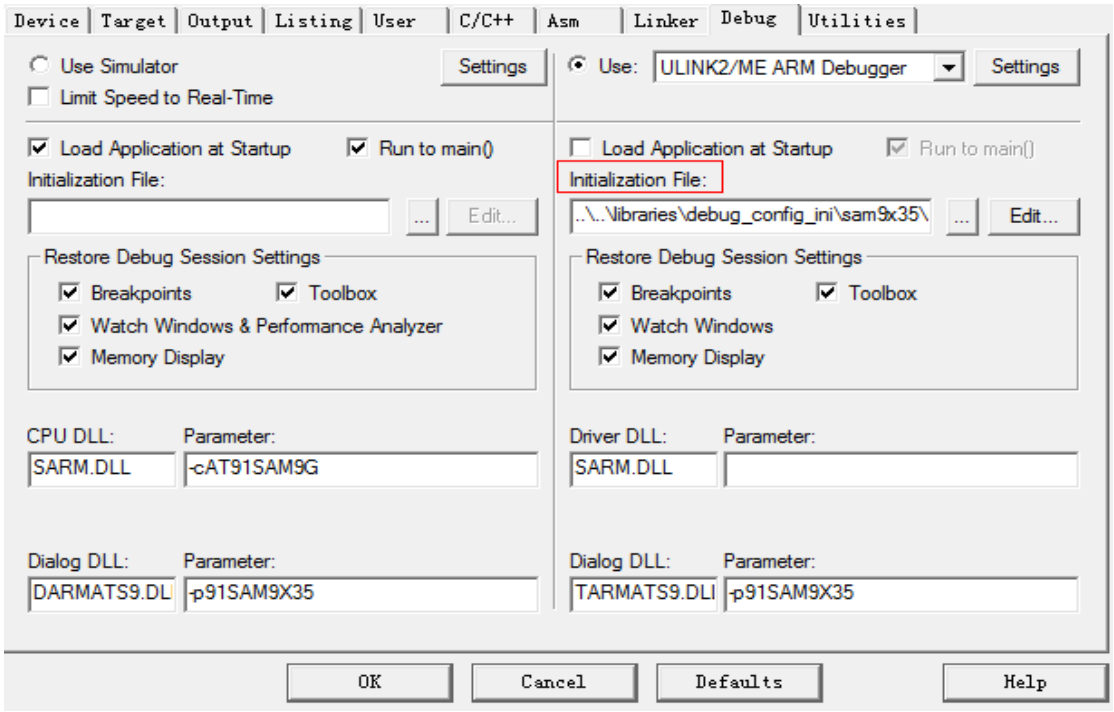
Analysis and Answers:

This problem occurs mainly caused by the following reasons:

(1) ULink2 connection or software does not recognize board

(2) Max JTAG Clock configuration is not right. In options menu "Options for target 'XXX' -> Debug-> 'ULINK2/ME Debugger' setting -> Max JTAG Clock", "Max JTAG Clock" should choose RTCK.

(3) Initialization File configuration has problem. In debug port, as shown below location:



The following information that appears in the command window shows ddram.axf load fails:

```
------------------------------ DDRAM configuration done ------------------------------
_MapRAMAt0();                  //* Set the RAM memory at 0x00300000 & 0x0000 0000
Changing mapping: RAM mapped to 0//_InitRSTC();
LOAD ddram\\ddram.axf INCREMENTAL
_____^
*** error 56, line 414: cannot open file
PC = 0x20000000;
//g,main
```

Please restore Initialization File in corresponding engineering as Initialization File.location:"04-MDK_Source\libraries\debug_config_ini",select 9g15/9g25/9g35/9x25/9x35 configuration file and Initialization file in Debug tab directly.

Appendix 2 sales FAQ and technical support

How to buy

We accept paypal payment and bank wire transfer

1. Paypal payment

Please select the products add into shopping cart, the checkout web page will redirect to paypal.com for you payment. Shipment fee will calculated automatically by your location region.

2. Bank wire transfer

Please email or fax us with products list you want, we will send you a pro-invoice with order value total, shipping cost and bank information.

Shipping details

Please select the shipping area catalogue for you location. If you have carrier account to pay the shipment fee, please select “Freight collect” and email us the carrier account. Please visit http://www.myirtech.com/support.asp for more details

Noted

1. The shipment will start in 3 biz days by Fedex Express, it usually take 7 days to reach regular cities or regions.
2. We will use DHL Express for West asia or middle east countries, it usually take 7 days to reach regular cities or regions.
3. The remote regions defined by Fedex/DHL may cause delay, 14 days in generally.
4. Some countries have strict import policy, we will help to make shipping invoice with you requirement, like invoice value, trade term, custom statements and H.S code etc. Please contact us with these shipment requirements if your country has strict custom affairs.

Support and maintains

MYIR provides 12 months warranty for hardware products if the defects or failures were not caused by wrong use.

Return steps for defective products

1. Please email or call us get a Return Merchandise Authorization (RMA) by providing purchase details and reasons for return (defective, incorrect etc).
2. MYIR will make a shipping invoice (list value total, item description etc) for you return request. China have strict limit on return products, so please use MYIR’s shipping invoice to return items to avoid custom delay.

Contact:

Tel:+86-0755-25622735 Fax: +86-0755-2553 2724

Mail to: sales@myirtech.com support@myirtech.com

Website: www.myirtech.com